1990

Problems to Test Parallel and Vector Languages -- II

John R. Rice
Purdue University, jrr@cs.purdue.edu

Jin Jing

Report Number:
90-1016

PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES - II

John R. Rice
Jin Jing

CSD-TR-1016
September 1990

Problems to Test Parallel and Vector Languages – II

John R. Rice**
and
Jin Jing

Computer Sciences Department
Purdue University
Technical Report CSD-TR-1016
CAPO Report CER-90-35
December, 1990

ABSTRACT

This report presents 17 problems selected to test the effectiveness of languages in expressing parallel and vector (and array) computations. Most problems have been extracted from larger computations and thus are somewhat artificial by themselves. However, they do represent a sampling of practical computations. Algorithms for the 17 problems are given in Fortran 77 and they are also designed to be used in timing computations.

This report is based on CSD-TR-516, May 1, 1985 with the same title. The problems are the same except that one has been added. In CDS-TR-516 the algorithms for the problems are expressed in four forms: Fortran 77, Fortran with Array and Parallel Extensions, PROTRAN with Extensions, and Cyber 205 Fortran. In the present report the algorithms are rewritten so as to (a) to correct some errors, (b) to make the coding style more uniform, (c) be parameterized, (d) to move I/O statements to the end, and (e) to improve the placement of runtime measuring statements. The algorithms are presented only in Fortran 77, it is easy to modify the other forms of CSD-TR-516 to conform to the new versions given here.

** This work supported in part by National Science Foundation grant CCR-8619817 and the Strategic Defense Initiative through Army Research Office contract DAAL03-90-G-00107.

INTRODUCTION

We present a set of 17 computations which naturally involve parallel and/or vector (or array) processing. The purpose is to provide a basis for testing the effectiveness of programming languages in expressing the parallel and array processing nature of computations. A secondary purpose is to provide a basis for measuring the execution efficiency (or speed up) of parallel and array processing systems.

We being by describing the changes made to the problems of the previous report [Rice. 1985] and providing some basic data on the new algorithms. Appendix A repeats the text of [Rice. 1985] with minor corrections and updates. Appendix B presents some additional remarks about the new problem 17. Appendix C presents the new algorithms in Fortran 77.

Changes Made in the Algorithms

A) Some of the previous algorithms had minor errors which prevented them from solving the stated problems even though the parallel and vector structures were correct.

B) The coding style has been made uniform.

C) Some of the previous algorithms were parameterized, but not all of them. Now they are all parameterized so that a wide range of computation sizes can be easily selected.

D) All of the input-output statements have been moved to the end of the algorithms.

E) The execution time statements have been placed so as to exclude timing the problem setup and output computations.

Problem 17

Problem 17 is an adaptive metalgorithm for numerical quadrature [Rice, 1975] which is naturally parallel. It is quite appropriate to test the effectiveness of parallel programming languages because of its dynamic, unpredictable behavior. See [Rice. 1976] for a detailed presentation of a specific instantiation of this metalgorithm and a proof of its correctness. It was intended to include it in the previous report [Rice, 1985] but a formulation could not be found which did not bias the implementation in some way. We were finally able to formulate it in a way that does not bias an implementation toward particular data structures and language constructs.

It is difficult to specify this algorithm at the Fortran 77 level without choosing specific data structures and synchronization mechanisms which bias the algorithm description toward some programming language styles or toward some parallel architecture. The use of a wide variety of data structures in this metalgorithm is discussed in [Rice, 1975]; the algorithm in [Rice, 1976] uses a queue to hold the intervals being processed and uses critical variables and sections to control access to the queue (synchronize the global information exchange).

The essence of this metalgorithm is that of a collection of items (intervals in this case) which are processed independently by the processors. The result of processing one item is the elimination of the input item and the creation of 0, 1 or 2 new items. New items are placed into the collection and the computation is continued until some objective is reached (the collection becomes empty in this case).

A different instance is presented here which suppresses somewhat the dependence on specific data structures and synchronization mechanisms. The earlier descriptions involve a single collection of unprocessed intervals which all processors had to access to insert and extract items. This is now replaced by having a collection for each processor and having the monitor processor move items from large local collections to small ones in order to keep the load balanced. There is still potential contention for access to the local collections when the monitor moves intervals from one processor to another to achieve load balancing. Some synchronization might be required here, but it is a much less important aspect of this algorithm. This instance of the metalgorithm seems not to bias it unduly toward or away from shared memory or distributed memory architectures.

METALGORITHM. The high level abstraction of the metalgorithm is exactly as in [Rice, 1975], that is:

CPU1 (HOST)	MAIN:	Initiates computation
CPU2(MONITOR)	MAIN:	Reads problems, Initialize control.
		Monitor computations for termination
CPU(IP)	MAIN:	Processes one interval
IP = 1 to NCPU	INIT:	Start one interval
	GET:	Obtain interval from the collection
	AREAS:	Compute numerical values
	PUT:	Obtain access to the collection
	INSERT:	Insert new intervals into collection.
		Update global control information

MORE SPECIFIC INSTANCE. The above metalgorithm is made more specific by specifying how the collection of intervals is managed. Each of the NCPU processors has its own collection which may be accessed by it and the monitor. The monitor has priority in accessing the collection, but we expect that the data structure provides for no conflict at all (and hence trivial synchronization cost). This can be accomplished using any of a number of data structures (e.g., stack, list, queue). One could consider implementations where efficiency is gained by allowing for conflict, and synchronizations, when the collection is nearly empty.

The collection is managed using the variable SIGNAL(IP) which can have one of three values:

"run"	:	normal mode of operation, the local collection is not large compared to others and has enough items to proceed.
"heavy"	:	the local collection is large and items are available to the monitor for redistribution to other processors
"starving"	:	the local collection is nearly empty and additional items are requested from the monitor.

These values are computed from four algorithm parameters

B_HIGH, B_LOW, C_HIGH, C_LOW

and the variables

COUNT(IP)	=	Number of items in processor IP's collection
TCOUNT	=	Total of COUNT(IP), IP = 1 to NCPU
BALANCE(IP)	=	NCPU * COUNT(IP) / TCOUNT

as follow:

```
IF ( BALANCE(IP) ≥ B_HIGH .AND. COUNT(IP) ≥ C_LOW )
      THEN SIGNAL(IP) = "heavy"
IF ( BALANCE(IP) ≤ B_LOW .AND. COUNT(IP) ≤ C_HIGH )
      THEN SIGNAL(IP) = "starving"
ELSE SIGNAL(IP) = "run"
```

Typical values for the parameters might be 2, 0.5, 3, and 4, respectively. We discuss load balancing in more detail in Appendix B for those who are really interested in adaptive quadrature.

This more specific instance is described in more detail by the actions of the processors as follows:

HOST_CPU :
1. Assign value of NCPU
2. Enables the other CPUs
3. Initializes all control variables

MONITOR_CPU :
1. Obtains problem data
2. Divides [A,B] into NCPU equal subintervals and initializes processor collections. Initializes variables (e.g., COUNT(IP), BOUND_ERROR, sets SIGNAL(IP) = "run") and enables all processors.
3. Monitors starving and, when detected, moves some intervals to a starving CPU so that it has BALANCE = 1.
4. Monitors BOUND_ERROR and terminates computations (with output) when BOUND_ERROR < EPS or when all interval collections are empty.

PROCESSOR(IP) : Invokes iteratively the sequence

INIT
GET
AREA1
PUT
INSERT

In a shared memory architecture, the variables COUNT(IP), SIGNAL(IP), BOUNDA(IP), etc. could be implemented by shared variables. Items could be moved between collections by changing pointers or by copying, depending on which is most efficient or convenient. In a DMMP (Distributed Memory, Message Passing) architecture the management of the collection would be implemented by asynchronous messages. The Fortran 77 program given here makes many specific choices for implementation, but it is not intended to specify the algorithm in further detail so that parallel language descriptions should only conform to the abstract metalgorithm presented here.

Performance Data

We tabulate the sequential execution times of these 17 problems for two sets of parameter values, one to give short execution times and one to give longer times.

Table 1. Execution times for Fortran 77 implementations of the computations on a SUN 3/50. Two selections of the parameters are given for each problem.

Problem	*Parameter Values and Execution Times*
1 :	WITH A,B,N = 0.000 1.000 10000 GIVES TN = 1.718554 TIME 10.8000 SECONDS, (10.5800 USER, 0.2200 SYSTEM)
	WITH A,B,N = 0.000 1.000 102400 GIVES TN = 1.718307 TIME 100.2400 SECONDS, (100.1800 USER, 0.0600 SYSTEM)
2 :	WITH N,M = 80 90 GIVES ESTAR = 446.482 TIME 9.3400 SECONDS, (8.9800 USER, 0.3600 SYSTEM)
	WITH N,M = 320 360 GIVES ESTAR = 1797.89 TIME 105.8200 SECONDS, (103.8600 USER, 1.9600 SYSTEM)
3 :	WITH N,M = 90 125 GIVES S = 91.13226 TIME 1.1667 SECONDS, (1.0833 USER, 0.0833 SYSTEM)
	WITH N,M = 270 625 GIVES S = 287.41537 TIME 6.7167 SECONDS, (3.7167 USER, 3.0000 SYSTEM)
4 :	WITH N = 64000 METHOD 1 GIVES 2.04807E+09 TIME 7.1200 SECONDS, (7.1000 USER, 0.0200 SYSTEM) METHOD 2 GIVES 2.04807E+09 TIME 6.8200 SECONDS, (6.8200 USER, 0.0000 SYSTEM) METHOD 3 GIVES 2.04807E+09 TIME 7.6600 SECONDS, (7.6000 USER, 0.0600 SYSTEM) METHOD 4 GIVES 2.04807E+09 TIME 6.8000 SECONDS, (6.8000 USER, 0.0000 SYSTEM) METHOD 5 GIVES 2.04807E+09 TIME 7.3400 SECONDS, (7.3400 USER, 0.0000 SYSTEM) METHOD 6 GIVES 2.04807E+09 TIME 7.2400 SECONDS, (7.2400 USER, 0.0000 SYSTEM) METHOD 7 GIVES 2.04807E+09 TIME 5.8600 SECONDS, (5.8600 USER, 0.0000 SYSTEM)
	WITH N = 320000 METHOD 1 GIVES 5.12007E+10 TIME 38.6000 SECONDS, (36.3800 USER, 2.2200 SYSTEM) METHOD 2 GIVES 5.12007E+10 TIME 38.2600 SECONDS, (35.9600 USER, 2.3000 SYSTEM) METHOD 3 GIVES 5.12007E+10 TIME 40.1000 SECONDS, (38.2400 USER, 1.8600 SYSTEM) METHOD 4 GIVES 5.12007E+10

TIME 37.6000 SECONDS. (35.2400 USER. 2.3600 SYSTEM)
METHOD 5 GIVES 5.12007E+10
TIME 36.8800 SECONDS. (36.1800 USER. 0.7000 SYSTEM)
METHOD 6 GIVES 5.12007E+10
TIME 36.2200 SECONDS. (36.0800 USER. 0.1400 SYSTEM)
METHOD 7 GIVES 5.12007E+10
TIME 28.6200 SECONDS. (28.6200 USER. 0.0000 SYSTEM)

---

5 : WITH NT, NS = 5 12
GIVES AVERAGE. GENIUS = 64.0874 F
AND LOW-ABOVE = 71.1111
TIME 97.2000 SECONDS. (96.8600 USER. 0.3400 SYSTEM)

WITH NT, NS = 50 1000
GIVES AVERAGE. GENIUS = 60.0098 F
AND LOW-ABOVE = 66.0109
TIME 107.5200 SECONDS. (105.6000 USER. 1.9200 SYSTEM)

---

6 : WITH N = 1024
GIVES SOLUTION = 495.352
TIME 11.6600 SECONDS. (11.6400 USER. 0.0200 SYSTEM)

WITH N = 16384
GIVES SOLUTION = 8165.01
TIME 210.0200 SECONDS. (208.7800 USER. 1.2400 SYSTEM)

---

7 : WITH N = 80
GIVES P(X) VALUES =
0.8225085E-01 0.1517956E-05 0.1138677E+05 NaN NaN
TIME 4.0200 SECONDS. (3.9800 USER. 0.0400 SYSTEM)

WITH N = 320
GIVES P(X) VALUES =
-NaN -NaN -NaN -NaN -NaN
TIME 43.3800 SECONDS. (43.2600 USER. 0.1200 SYSTEM)

---

8 : WITH N,M = 781 20
GIVES SUM OF DIFFERENCE TABLE = -4.30577
TIME 5.7600 SECONDS. (5.6800 USER. 0.0800 SYSTEM)

WITH N,M = 3161 40
GIVES SUM OF DIFFERENCE TABLE = -4.75962
TIME 34.9000 SECONDS. (34.6600 USER. 0.2400 SYSTEM)

---

9 : WITH N,M = 80 60
GIVES SUM = 1.06241E+07
TIME 2.0400 SECONDS. (2.0000 USER. 0.0400 SYSTEM)

WITH N,M = 200 180
GIVES SUM = 4.87207E+08
TIME 15.1600 SECONDS. (15.1000 USER. 0.0600 SYSTEM)

---

10 : WITH N = 40
SUM OF ELEMENTS AND SUM OF DIAGONAL =

```
        0.348794E+06  0.434005E+01
TIME   4.8400 SECONDS, (  4.8400 USER,   0.0000 SYSTEM)

WITH N =  100
SUM OF ELEMENTS AND SUM OF DIAGONAL =
        0.817687E+38  0.817687E+38
TIME  49.7600 SECONDS, ( 49.7000 USER,   0.0600 SYSTEM)

11 :  WITH N =  1000
      GIVEN FMOM(N) (N=1,4) =  0.1903  0.1882  0.1820  0.1720
      TIME   5.2800 SECONDS, (  5.2200 USER,   0.0600 SYSTEM)

      WITH N =  5000
      GIVEN FMOM(N) (N=1,4) =  0.0381  0.0381  0.0380  0.0379
      TIME  26.6000 SECONDS, ( 26.4200 USER,   0.1800 SYSTEM)

12 :  WITH N,M =  200   200
      GIVES CORNER PRODUCTS =  0.  -7.99980E+06
      TIME   2.0800 SECONDS, (  1.8600 USER,   0.2200 SYSTEM)

      WITH N,M =  400   400
      GIVES CORNER PRODUCTS =  0.  -6.39996E+07
      TIME   7.6400 SECONDS, (  7.1200 USER,   0.5200 SYSTEM)

13 :  WITH N =  1000
      GIVES E =   5.16012E+06
      TIME  14.4000 SECONDS, ( 14.3000 USER,   0.1000 SYSTEM)

      WITH N =  10000
      GIVES E =   5.49776E+09
      TIME 142.8000 SECONDS, (142.4600 USER,   0.3400 SYSTEM)

14 :  FUNCTION  1  ON   0.00  1.00  HAS TRUE =  1.718282
      EVALUATION WITH  500 INTERVALS, L =  1
      METHOD   ANSWER      NO OF POINTS    ERROR      LOG ERROR
      1   1.718282341957   501.  -0.000000476837  -.632163000107E+01
      2   1.718281865120   501.   0.000000000000  -.310000000000E+02
      3   1.301083683968   498.   0.417198181152  -.379657596350E+00

      FUNCTION  2  ON   0.00  1.00  HAS TRUE =  0.522210
      EVALUATION WITH  500 INTERVALS, L =  1
      METHOD   ANSWER      NO OF POINTS    ERROR      LOG ERROR
      1   0.522214233875   501.  -0.000004291534  -.536738729477E+01
      2   0.522206485271   501.   0.000003457069  -.546129179001E+01
      3   0.333972334862   498.   0.188237607479  -.725293576717E+00

      FUNCTION  3  ON  -1.00  2.00  HAS TRUE =  6.299197
      EVALUATION WITH  500 INTERVALS, L =  1
      METHOD   ANSWER      NO OF POINTS    ERROR      LOG ERROR
      1   6.299215793610   501.  -0.000019073486  -.471956968307E+01
      2   6.299195766449   501.   0.000000953674  -.602059984207E+01
      3   4.306991100311   498.   1.992205619812  0.299334168434E+00

      TIME   2.8400 SECONDS, (  2.7800 USER,   0.0600 SYSTEM)
```

FUNCTION 1 ON 0.00 1.00 HAS TRUE = 1.718282
EVALUATION WITH 2000 INTERVALS, L = 1

METHOD	ANSWER	NO OF POINTS	ERROR	LOG ERROR
1	1.718281865120	2001.	0.000000000000	-.310000000000E+02
2	1.718282580376	2001.	-0.000000715256	-.614553880692E+01
3	1.298349142075	1998.	0.419932723045	-.376820296049E+00

FUNCTION 2 ON 0.00 1.00 HAS TRUE = 0.522210
EVALUATION WITH 2000 INTERVALS, L = 1

METHOD	ANSWER	NO OF POINTS	ERROR	LOG ERROR
1	0.522205352783	2001.	0.000004589558	-.533822917938E+01
2	0.522202551365	2001.	0.000007390976	-.513129806519E+01
3	0.333862751722	1998.	0.188347190619	-.725040853024E+00

FUNCTION 3 ON -1.00 2.00 HAS TRUE = 6.299197
EVALUATION WITH 2000 INTERVALS, L = 1

METHOD	ANSWER	NO OF POINTS	ERROR	LOG ERROR
1	6.299193859100	2001.	0.000002861023	-.554347848892E+01
2	6.299199581146	2001.	-0.000002861023	-.554347848892E+01
3	4.315896987915	1998.	1.983299732208	0.297388345003E+00

TIME 10.8800 SECONDS, (10.7400 USER, 0.1400 SYSTEM)

---

15 : IN EQUALLY SPACED POINTS
WITH N = 10
GIVEN MAX ERROR AND DECAY EXPONENT =
0.1490E-07 -2.12

IN CHEBYSHEV SPACED POINTS
WITH N = 10
GIVEN MAX ERROR AND DECAY EXPONENT =
0.4098E-07 -4.13
TIME 13.3000 SECONDS(USER 13.2800 SYSTEM 0.2400)

IN EQUALLY SPACED POINTS
WITH N = 20
GIVEN MAX ERROR AND DECAY EXPONENT =
0.1118E-07 7.90

IN CHEBYSHEV SPACED POINTS
WITH N = 20
GIVEN MAX ERROR AND DECAY EXPONENT =
0.7451E-08 0.00
TIME 79.8000 SECONDS, (79.5000 USER, 0.5800 SYSTEM)

---

16 : WITH N = 4
RESIDUE 1 := 0.269739825853E-05
RESIDUE 2 := 0.190734863281E-05
RESIDUE 3 := 0.363490607924E-05
RESIDUE 4 := 0.745058059692E-07
TIME 0.0600 SECONDS, (0.0600 USER, 0.0000 SYSTEM)

WITH N = 20

```
      *** THE CONDITION NUMBER IS TOO HIGH
      TIME 0.6800 SECONDS, (0.6600 USER,   0.0200 SYSTEM)
```

```
17 :  WITH A,B,EPS = 0.00010000  1.00010000     0.01000000
      GIVES THE AREA = 9.21281624 AND THE BOUND = 0.00949743
      TIME   1.8800 SECONDS, (  1.8200 USER,   0.0600 SYSTEM)

      WITH A,B,EPS = 0.000100001.00010002     0.00010000
      GIVES THE AREA = 9.21067333 AND THE BOUND = 0.00009804
      TIME  16.6600 SECONDS, ( 16.4800 USER,   0.1800 SYSTEM)
```

REFERENCES

J.R. Rice, A metalgorithm for adaptive quadrature, *J. Assoc. Comp. Mach.*, 22 (1975), 61-82.

J.R. Rice, Parallel algorithms for adaptive quadrature III – Program correctness, *ACM Trans. Math. Software*, 2 (1976), 1-30.

J.R. Rice, Problems to test parallel and vector languages, CSD-TR-516, Computer Science, Purdue University, May 1, 1985.

APPENDIX A: CSD-TR-516, May 1, 1985.

PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES

John R. Rice

A-1. INTRODUCTION

A set of 16 problems is presented whose purpose is to test the effectiveness of programming languages in expressing parallel and vector (or array) computations. Most of the problems have been abstracted from common, realistic programs; a few problems are from small, complete programs. These problems are useful to provide an independent and somewhat uniform means to test the many new programming languages being proposed for vector and parallel computation.

A-1.A. The Forms of the Problems

The problems ae presented in four forms:

Form A: Fortran 77. This is an ordinary sequential form which serves to define the computation precisely. This form is complete with timing code and output that can be used to check other implementations.

Form B: Fortran with Array and Parallel Extensions. This is an extension which is not precisely defined, but which resembles Fortran 90 in many ways. However, it also differs in some significant ways and it includes constructs for parallel computation.

Form C: PROTRAN with Extensions. PROTRAN is a Fortran extension developed by IMSL which has considered vector/matrix capabilities as well as high level, problem solving statements. The extension used here (but not precisely defined) completes the vector capabilities to a level comparable with Form B (and Fortran 90). It also includes parallel constructs.

Form D: Cyber 205 Fortran. This is CDC's Fortran extension especially targeted for the Cyber 205 vector computer. The Fortran 77 codes were used as a starting point and then a reasonable effort was put into optimizing execution time. In other words, the codes were hand vectorized.

Many of the programs are parameterized in some simple way to allow one to change the problem "size".

A-1.B. Principal Observations

This effort arose in the early 1980's from considering the Fortran 90 proposals and from attempting to evaluate them. The problem set was enlarged to consider also the nature of truly parallel computations expressed in Fortran-like languages.

Studying these 16 problems has led to the following subjective conclusions:

1. **Array Notations.** Science has evolved notation for vectors and arrays which mixes the linear algebra notation (e.g., Solve $\mathbf{Ax} = \mathbf{b}$) with identical notations (e.g., where $A = \{a_{ij}\} = \{1 / (i + j - 1)$ for $i \le j, \quad j \le m\}$ and $b = \{b_j\} = \{1$ for $j = 1, 0$ otherwise$\}$). Both forms of the notation are very natural and must be included in some way if the language is to avoid clumsy, error prone constructions.
2. **Array Operations.** A substantial number of arrays operators beyond simple arithmetic are in common scientific use (e.g., sum, product, transpose, inverse, matrix product) and should be included in the language as naturally as possible. Further, new operators to create and manipulate arrays are also needed (e.g., concatenate two vectors, add a new row to an array). It is inevitable that there be numerous operators and that most of them are implemented as procedures (functions or subroutines). The natural use of the SUM, PRODUCT, MAX and MIN operators is difficult to achieve without substantial changes in the usual Fortran syntax.
3. **Language Level.** Programming productivity and execution efficiency both require that the language has the power to express computations at the natural level of scientific notation. It is an inexcusable waste of talent to ask all (or even many) scientists to learn artificial rules about DO-loop organization and

similar machine/compiler effects. The vector languages in current use from Cray Research, CDC and Los Alamos can only be justified as temporary expediencies.

4. **Parallel Computation.** Languages constructs for parallel computation are still very open. We observe:

   4a. *Control constructs.* It is easy to see that only a few control constructs are needed and that there are several reasonable alternatives for them.

   4b. *Data access and control.* A principal difficulty (which does not show up on toy programs of one page) is how to organize and access data. The Fortran COMMON and parameter passing lead to obvious data synchronization problems.

The remainder of this report describes the problem set, first in a summary way using tables of characteristics such as lengths and execution timings. Then there is a set of brief descriptions (only a few lines) followed by four appendices which give all the problems in each of the four forms.

A-2. CHARACTERISTICS OF THE PROBLEMS

This problem set evolved from consideration of the vector/array features that have been proposed for the next Fortran standard. papers and reports that discuss these extensions along with examples are [Smith, 1982], [Wilson, 1982], [Rice, 1981], [Rice, 1984]. A Few of these problems were constructed just for this report. A few are taken from the book [Rice, 1983], these are programs that exhibit a high degree of parallelism.

The Fortran 77 form of the programs use the following general design criteria.

1. The computational "size" of the problem can be varied easily. This is implemented with PARAMETER and DATA statements.
2. Timing is included. This involves the availability of a system dependent timing routine.
3. A simple numerical value is printed to provide a check of the accuracy of alternative implementations.

The PROTRAN and extended Fortran forms are the more readable and they also include more general comments describing the problem.

Table A-1 provides a simple summary of the characteristics of the computations of the problems. The following terminology is used:

vector: – means one-dimensional array manipulation of normal mathematical vectors (in the linear algebra sense).

array: – means manipulation of arrays that are not matrices in the linear algebra sense. The dimension of the arrays involved is given in the table.

matrix: – means manipulation of matrices and vectors in the linear algebra sense.

parallel: – means independent computations that are not obviously recast in the form of array manipulation.

formula: – means the problem involves computations that can be expressed naturally in terms of common mathematical notations.

procedures: – means the problem involves a procedure or function that interacts with parallelism or vectorization. Simple procedures like SUM, DOTPRODUCT, COS, etc., are not included.

The problems are relatively small by nature. Table A-2 gives the lengths of each of the forms of the problems. Note that the design of the problems requires several declarations, timing and I/O statements which would not be present in the "production" version of these computations. The counts in Table A-2 do not include these "extra" lines nor comments: they represent the computational kernel of the problem.

Table A-1. Six characteristics of the problems, see text for definitions of the headings.

Problem	Vector	Array	Matrix	Parallel	Formula	Procedures
1		1			x	x
2		2			x	
3		2			x	
4		1			x	
5		2				
6	x		x			
7		1			x	x
8		2				
9		3				
10	x		x			
11		1			x	
12		2				
13		1		x		
13H		1		x	x	
14		1		x	x	x
15		2		x	x	x
16	x		x	x	x	
17				x		x

Table A-2. Counts of the lines of the computational kernel in the four forms of each problem.

Form		Problem 1	2	3	4	5	6
A.	Fortran 77	5	11	10	39	41	26
B.	Extended Fortran	3	7	6	21	20	28
C.	Extended PROTRAN	4	5	4	19	15	24
D.	205 Fortran (Opt)	7	7	4	24	35	24
	205 Fortran (seq)	5	16	10	40	42	26
		7	8	9	10	11	12
A.	Fortran 77	28	12	29	28	12	19
B.	Extended Fortran	22	14	29	18	9	19
C.	Extended PROTRAN	11	12	24	21	8	15
D.	205 Fortran (Opt)	18	12	25	21	20	18
	205 Fortran (seq)	28	12	28	28	12	19
		13	13H	14	15	16	
A.	Fortran 77	13	21	72	64	60	
B.	Extended Fortran	16	21	64	52	116	
C.	Extended PROTRAN	13	17	63	56	27	
D.	205 Fortran (Opt)	13	21	97	112	62	
	205 Fortran (seq)	13	21	762	67	60	

Table A-2 shows two variations of the Cyber 205 Fortran forms. The *205 Fortran (Opt)* refers to the hand optimized version, the one expected to be used in a production code. The *205 Fortran (Seq)* refers to the Fortran 77 form with minimum modifications needed to make it run on the Cyber 205. The changes made are due to different rules about the use of indexes in arrays, etc.

Table A-3 shows some run time data for the three executable forms of the problems (Fortran 77, 205 Fortran (Opt) and 205 Fortran (Seq)). We do not analyze in any detail the data in this problem; we do note that there is some erratic behavior.

A-3. BRIEF DESCRIPTIONS OF THE PROBLEMS

In this section we give a brief description of the 16 problems. The precise descriptions are the Fortran 77 forms of the problem.

Table A-3. Execution times of three versions of the problems. The Fortran 77 version is run on the VAX 11/780 (single precision, F77 compiler) and the other two on the Cyber 205 (single precision, Fortran 200 compiler). The second column has the ratio of the column 1 to column 2 times under its entries. Similarly, the column 2 to columns 3 ratios are given under the column 3 entries.

Execution Time in Seconds

Problem	Fortran 77	205 Fortran (Seq)	205 Fortran (Opt)
1	0.017	0.0135	0.0127
		1.3	1.06
2 (5 cases)	1.177	0.0455	0.0459
		25.9	0.99
3 (3 cases)	0.067	0.0028	0.0131
		23.9	0.21
4 (7 cases)	0.250	0.032	0.0210
		7.8	1.5
5 (3 cases)	2.617	0.0758	0.0328
		34.5	2.3
6 (2 cases)	2.417	0.0411	0.0283
		58.8	1.45
7 (3 cases)	0.383	0.0345	0.0299
		11.1	1.15
8 (3 cases)	0.250	0.0220	0.0197
		11.4	1.12
9 (3 cases)	0.583	0.0254	0.0219
		23.0	1.16
10 (3 cases)	0.250	0.0192	0.0181
		13.	1.06
11 (2 cases)	1.033	0.0383	0.0160

		27.0	2.39
12 (2 cases)	0.083	0.0144	0.0133
		5.8	1.08
13	0.467	0.0218	0.0132
		21.4	1.65
14	8.317	0.3367	0.0616
		24.7	5.46
15	16.917	0.8704	0.9349
		19.4	0.93
16	0.400	0.0379	0.0338
		10.6	1.12

Problem 1: evaluate the trapezoidal rule estimate of an integral of $f(x)$:

$$T_N = h^*(f(a)/2 + \sum_{i=1}^{N-1} f(a + ih) + f(b)/2)$$

Problem 2: Compute the value of

$$e^* = \sum_{i=1}^{n} \prod_{j=1}^{m} (1. + e^{(-1|i-j|)})$$

Problem 3: Compute the value of $S = \sum_{j=1}^{n} \prod_{i=1}^{m} a_{ij}$

Problem 4: Compute the value of $R = \sum_{\substack{j=1 \\ x_i \neq 0}} \frac{1}{x_i}$

Problem 5: One has a table of the i-th student's score on the j-th test. One is to:

(a) list the top score for each student = top_i
(b) give the number of scores above the average = NABOVE
(c) increase all the above average scores by 10 percent
(d) give the lowest score that is above average = LOW_ABOVE
(e) say whether any student has all scores above average = GENIUS

Problem 6: Solve the tridiagonal system $Tx = y$ by the special, vector oriented algorithm of [Jordan, 1979]. The matrix T is represented by L, D and U, its lower diagonal, main diagonal and upper diagonal.

Problem 7: Compute polynomial interpolant values of $f(x)$ at five points using Lagrange interpolation formulas:

$$p(x) = \sum_{i=1}^{N} f(x_i) l_i(x) \qquad l_i(x) = \prod_{\substack{j=1 \\ j \neq i}} (x - x_j) \Big/ \prod_{\substack{j=1 \\ j \neq i}} (x_i - x_j)$$

Problem 8: The divided difference table for a set of data x_i, $y_i = f(x_i)$ is defined by the formulas

$$f[x_i] = y_i$$

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

The problem is to compute the first M columns of the divided difference table

$$D_{ik} = f[x_i, x_{i+1}, \ldots, x_{i+k-1}]$$

Problem 9: One has an array u_{ij} of values on the N by M grid and wants to replace each value by the average of its value plus those of all its neighbors. This is expressed by

$$u_{ij} = \Big(\sum_{Neighbors} u_{ij} \Big) / (Number\ of\ neighbors)$$

This computation is typical of what one does in solving partial differential equations, image processing and geometric modeling.

Problem 10: Computer the LU factorization of the N by N matrix $A = a_{ij}$ using Gauss elimination with pivoting.

Problem 11: Read sets of data d_i, $i = 1, \ldots, N$, trim the negative values to zero and large values to 1000, do a logarithmic transformation $d_i = \log(1 + d_i)$ and compute the first four Fourier moments $\sum_{i=1}^{N} d_i \cos(\pi i / (N + 1))$ then save these moments and the data ID in a data base.

Problem 12: Give the m by m matrix A, the 1 by m vector R, the m by 1 vector C and a number a, construct the array

$$ABIG = \begin{bmatrix} A & C \\ R & a \end{bmatrix}$$

Problem 13: For given vectors a, b, c and d of dimension N, compute the new vector

$a_i = a_i \sin b^i$

If $a_i < \cos(c_i)$ then $a_i = a_i + c_i$

else $a_i = a_i - d_i$

and then compute

$$e = \sum_{j=1}^{N} a_i^2$$

Problem 13H: Modify Problem 13 for a machine (e.g., such as the Denelcor HEP) that wants to have the computation split into groups of 20 processes.

Problem 14: Carry out a test of four methods to integrate three different functions with 10 different levels of accuracy each. Print out a table with all the results including the number of function evaluations used in each integration. This problem comes from [Rice, 1983], page 204.

Problem 15: Carry out a comparison of two types of interpolation points (equispaced and Chebyshev spaced) for Hermite interpolation using piecewise cubic polynomials The interpolant's value v at y can be expressed as

$$v(y) = \sum_{j=1}^{N} f(x_j)\, h_{1j}\,(y) + f'\,(x_j)\, h_{2j}\,(y)$$

Problem 16: Solve a matrix equation $Ax = B$ where A is an N by N Hilbert matrix and BR is an N by 4 matrix. The matrix order N takes on the values 4, 8, 12, 16 and 20 and the B column-vectors are, respectively, the first column of the identity matrix, all 1's, and 0.01 random perturbation of all 1's, and alternating +1, −1.

A-4. BRIEF DESCRIPTIONS OF THE LANGUAGES

The Fortran 77 form is in Fortran as standardized in 1977.

The Extended Fortran form is similar to Fortran 90. The features used here which are taken directly for proposals adopted in Fortran 90 include:

Array assignments and expressions

Array oriented functions, e.g., SUM, PRODUCT, COUNT, MASTK_SUM

The FORALL statement (which was deleted from Fortran 90 in the end) is also used as in

```
FORALL (I = 1:N) A(I) = A(I - 1) / (I**2 + 2)
```

Note that the syntax and form used here is from 1985 documents and the final form of Fortran 90 might be different. Two array statements are used here which are not

included in Fortran 90. They are

1. Block FORALL. e.g.,

```
FORALL (I = 1:N)
  A(I) = B(I) + C(I)
  CALL GETIT(D,I)
END ALL
```

2. Range variables. This allows the working size of arrays to be independent of the storage allocation. See [Rice, 1982a] for more details.

Finally, there are statements which are specifically for parallel computation, namely

3. DO PARALLEL, specify a block of independent statements:

```
DO PARALLEL
  CALL SUB1(A,B)
  CALL SUB2(C,D)
END PARALLEL
DO PARALLEL(I = 1,N)
  CALL ADDFUNK(F,G,I)
END PARALLEL
```

4. CRITICAL, declares a variable which might be accessed simultaneously by different codes and where single access is to be provided.
5. BEGIN-END with declarations. This provides two key facilities:
   (A) simple grouping of statements to indicate blocks to be handled in parallel,
   (B) declarations of local (dynamic) arrays and variables within blocks. Consider

```
DO PARALLEL
BEGIN
```

```
COMMON / SELECT / JFUNK
DO PARALLEL(I - 1,N)
  BEGIN
    REAL  H, TRAP, FI(I)
    H  = (B - A) / (I + 1)
    FORALL  (J = 1,I) FI(J) = F(A + J*H)
    TRAP  = F(A) + F(B) + DOTPRODUCT(FI,WT)
  END
  END PARALLEL
END
  more code
END PARALLEL
```

This code creates a COMMON block SELECT for the first group of statements. Then *N* groups of statements are created within the first group, each with its own values of *H*, TRAP and array FI. The statements within a BEGIN-END block are executed sequentially.

For more information use the X3J3 documents describing the Fortran 8X proposals as well as [Smith, 1982], [Wilson, 1982], [Rice, 1982b] and [Rice, 1984].

PROTRAN is described in Chapter 15 of [Rice, 1983], [Aird and Rice, 1983]. PROTRAN is a higher level language for mathematical and scientific computation and, as such, has vector and matrices (not 2-D arrays) as data types. It also has operators like SUM and PRODUCT. The extension used here is in two parts. First, some of the vector and array facilities of Fortran 8X are included so that PROTRAN is as suitable for vector processing as Fortran 8X. This is a small extension. Second, the parallel constructs used for the extension of Fortran are also included, that is:

DO PARALLEL

CRITICAL

BEGIN-END with declaration

The Cyber 205 Fortran, called Fortran 200, is documented in [CDC, 1983]. It has two types of extensions. First are vector processing facilities similar to many of those included in the extended Fortran and extended PROTRAN. They use somewhat different syntax and are more limited due to the specific nature of vectors on the Cyber 205. Second are machine specific subroutines called the Q8 subroutines. These calls allow one to insert specific machine language statements into the object code generated by the Fortran compiler. They are often needed to obtain maximum performance from the Cyber 205.

A-5. REFERENCES

Aird, T.J. and J.R. Rice, (1983), PROTRAN: Problem solving software. *Adv. Engin. Software*, 5, pp. 202-206.

CDC (1983), Fortran 200 Reference Manual.

Jordan, Tom (1979), private communication.

Rice, John R., (1981), Array facilities in programming languages, Computer Science Department, CSD-TR-380, Purdue University, October, 37 pages.

Rice, John R., (1982a), A proposal for range variables in Fortran 8X, unpublished report, 5 pages.

Rice, John R., (1982b), Array facilities in Fortran, unpublished report, 28 pages.

Rice, John R., (1983), *Numerical Methods, Software and Analysis*, McGraw-Hill.

Rice, John R., (1984), Fortran extensions for parallel and vector computations, Computer Science Department, CSD-TR-470, Purdue University, January, 8 pages.

Smith, Brian T., (1982), Array processing features in the next Fortran. In *The Relationship between Numerical Computation and Programming Languages* (J. Reid, ed.), North-Holland, pp. 163-177.

Wilson, Alan, (1982), Examples of Array Processing int he next Fortran. In *The Relationship between Numerical Computation and Programming Languages* (J. Reid, ed.), North-Holland, pp. 179-182.

APPENDIX B: LOAD BALANCING FOR ADAPTIVE QUADRATURE

The load balancing presented for Problem 17 is simple, but probably not satisfactory for a production quality quadrature code. The objective is to distribute the work evenly over the processors and work is not simply related to the number of intervals. One should consider two extreme cases

1. *The integrand $f(x)$ is uniformly smooth.* In this case one expects the intervals to be cut repeatedly until a certain size is reached. At that time all of them are processed, BOUND_ERR is less than EPS, and the computation ceases. The load should remain evenly distributed over all processors provided (a) if was evenly distributed to start, (b) the processors are equally quick in processing an interval.
2. *The integrand $f(x)$ has one or a few singularities.* In this case there is an initial phase where most of the integration is complete – as measured by the length of the intervals involved. However, the few very short intervals left will continue to require lengthy processing. Thus, each singularity becomes a source of a long sequence of shorter and shorter intervals. Load balancing is critical for good performance.

We suggest two changes to make a program more efficient.

A. **Subdivide Intervals into Many Parts.** One should divide each interval into a larger and larger number of subintervals as the load starts to become unbalanced. In extreme cases, one probably should have the order of p subintervals when using p processors.

B. **Measure Work by Both Intervals Counts and Lengths.** Modify the previous collection management by renaming BALANCE(I) to be

BALANCE_COUNT(IP) = NCPU * COUNT(IP) / TCOUNT.

Then introduce

LENGTH(IP)	=	Length of intervals in IP's collection
TLENGTH	=	Total of LENGTH(IP), IP = 1 to NCPU
BALANCE_LENGTH(IP)	=	NCPU * LENGTH(IP) / TLENGTH.

We introduce a weight α as follows

$$\begin{aligned} AB &= \text{Original interval length} \\ \alpha &= TLENGTH / AB \end{aligned}$$

and define BALANCE(IP) dynamically as

$$BALANCE(IP) = \alpha * BALANCE\_LENGTH + (1 - \alpha) * BALANCE\_COUNT$$

We believe that these changes would make the algorithm much more practical.


```
C
C     PROBLEM 1
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (KASES=11)
      DIMENSION TN(KASES)
      DIMENSION TARRAY(2)
      INTEGER N(KASES)
      DATA A,B / 0.0,1.0 /
      DATA N / 100, 200, 400, 800, 1600, 3200,
     X         6400, 12800, 25600, 51200, 102400 /
C
C              LOOP OVER KASES
C
      TIME1 = DTIME(TARRAY)
      DO 20 K = 1, KASES
         H = (B-A)/(N(K)-1)
         SUM = 0.0
         DO 10 I = 1, N(K)-1
            SUM = SUM+F(A+H*I)
   10    CONTINUE
         TN(K) = H*((F(A)+F(B))/2.+SUM)
   20 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 50 K = 1, KASES
         PRINT 30,A,B,N(K)
   30    FORMAT ('PROBLEM 1 WITH A,B,N = ',2F10.3,5X,I6)
         PRINT 40,TN(K)
   40    FORMAT ('GIVES TN =',F10.6)
   50 CONTINUE
      PRINT 60,TIME2,TARRAY(1),TARRAY(2)
   60 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END
C
      FUNCTION F (X)
      F = EXP(X)
      RETURN
      END
```

```
C
C     PROBLEM 2
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (NDIM=320,MDIM=360,KASES=6)
      DIMENSION TEMP(NDIM,MDIM),PROD(MDIM),TESTAR(KASES)
      INTEGER N(KASES),M(KASES)
      DIMENSION TARRAY(2)
      DATA N / 10,20,40,80,160,320 /
      DATA M / 15,30,60,90,180,360 /
C
C               LOOP OVER KASES
C
      TIME1 = DTIME(TARRAY)
      DO 50 K = 1, KASES
         DO 30 I = 1, N(K)
            DO 10 J = 1, M(K)
               T1 = FLOAT(-IABS(I-J))
               TEMP(I,J) = 1+EXP(T1)
   10       CONTINUE
            PROD(I) = 1
            DO 20 J = 1, M(K)
               PROD(I) = PROD(I)*TEMP(I,J)
   20       CONTINUE
   30    CONTINUE
         ESTAR = 0
         DO 40 I = 1, N(K)
            ESTAR = ESTAR+PROD(I)
   40    CONTINUE
         TESTAR(K) = ESTAR
   50 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 70 K = 1, KASES
         PRINT 60,N(K),M(K)
   60    FORMAT ('PROBLEM 2 WITH N,M =',I6,2X,I6)
         PRINT *,'GIVES ESTAR =',TESTAR(K)
   70 CONTINUE
      PRINT 80,TIME2,TARRAY(1),TARRAY(2)
   80 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      END
```

```
C
C     PROBLEM 3
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      INTEGER SIZE
      PARAMETER (NDIM=270,MDIM=625,SIZE=NDIM*MDIM,KASES=4)
      DIMENSION A(NDIM,MDIM),P(NDIM),N(KASES),M(KASES),TS(KASES)
      DIMENSION TARRAY(2)
      DATA A / SIZE*1.0001 /,N / 10,30,90,270 /
      DATA M / 5,25,125,625 /
      TIME1 = DTIME(TARRAY)
      DO 30 K = 1, KASES
         NN = N(K)
         MM = M(K)
         DO 10 I = 1, NN
            P(I) = 1
            DO 10 J = 1, MM
               P(I) = P(I)*A(I,J)
   10    CONTINUE
         S = 0
         DO 20 I = 1, NN
            S = S+P(I)
   20    CONTINUE
         TS(K) = S
   30 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 60 K = 1, KASES
         PRINT 40, N(K),M(K)
   40    FORMAT('PROBLEM 3 FOR N,M =',I6,2X,I6)
         PRINT 50, TS(K)
   50    FORMAT('GIVES S =',F10.5)
   60 CONTINUE
      PRINT 70,TIME2,TARRAY(1),TARRAY(2)
   70 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      END
```

```
C
C     PROBLEM 4
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (MDIM=64000,KASES=3)
      DIMENSION X(MDIM),Y(MDIM)
      DIMENSION N(KASES),TARRAY(2)
      DATA N / 1000,8000,64000 /
      DO 130 K = 1, KASES
         NDIM = N(K)
         PRINT *,' '
         PRINT *,'PROBLEM 4 WITH N =',N(K)
         DO 10 I = 1, NDIM
            X(I) = 1./(1.+I)
   10    CONTINUE
         X(11) = 0.0
         Y(11) = 0.0
         TIME1 = DTIME(TARRAY)
C
C                METHOD 1
C
         DO 20 I = 1, NDIM
            IF (X(I).NE.0) THEN
               Y(I) = 1./X(I)
            ENDIF
   20    CONTINUE
         R = 0
         DO 30 I = 1, NDIM
            R = R+Y(I)
   30    CONTINUE
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 1 GIVES',R
         TOTALTIME = TOTALTIME+TIME2
         PRINT 40,TIME2,TARRAY(1),TARRAY(2)
   40    FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,
     *      ' SYSTEM)')
C
C                METHOD 2
C
         TIME1 = DTIME(TARRAY)
         DO 50 I = 1, NDIM
            IF (X(I).NE.0) THEN
               Y(I) = 1./X(I)
            ENDIF
   50    CONTINUE
         R = 0
         DO 60 I = 1, NDIM
            R = R+Y(I)
   60    CONTINUE
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 2 GIVES',R
         TOTALTIME = TOTALTIME+TIME2
         PRINT 40,TIME2,TARRAY(1),TARRAY(2)
C
C                METHOD 3
C
         TIME1 = DTIME(TARRAY)
         SUM = 0.0
         DO 70 I = 1, NDIM
            IF (X(I).NE.0) Y(I) = 1./X(I)
            SUM = SUM+Y(I)
   70    CONTINUE
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 3 GIVES',SUM
         TOTALTIME = TOTALTIME+TIME2
         PRINT 40,TIME2,TARRAY(1),TARRAY(2)
C
C                METHOD 4
C
         TIME1 = DTIME(TARRAY)
         DO 80 I = 1, NDIM
            IF (X(I).NE.0) THEN
               Y(I) = 1./X(I)
            ENDIF
   80    CONTINUE
         R = 0
         DO 90 I = 1, NDIM
            R = R+Y(I)
   90    CONTINUE
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 4 GIVES',R
         TOTALTIME = TOTALTIME+TIME2
         PRINT 40,TIME2,TARRAY(1),TARRAY(2)
C
C                METHOD 5
C
         TIME1 = DTIME(TARRAY)
         SUM = 0.0
         DO 100 I = 1, NDIM
            IF (X(I).NE.0) THEN
               SUM = SUM+1./X(I)
            ENDIF
  100    CONTINUE
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 5 GIVES',SUM
         TOTALTIME = TOTALTIME+TIME2
         PRINT 40,TIME2,TARRAY(1),TARRAY(2)
C
C                METHOD 6
C
         TIME1 = DTIME(TARRAY)
         R = 0
         DO 110 I = 1, NDIM
            IF (X(I).NE.0.) THEN
               R = R+1./X(I)
            ENDIF
  110    CONTINUE
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 6 GIVES',R
         TOTALTIME = TOTALTIME+TIME2
         PRINT 40,TIME2,TARRAY(1),TARRAY(2)
C
C                METHOD 7
C
         TIME1 = DTIME(TARRAY)
         X(11) = 1
         R = 0
         DO 120 I = 1, NDIM
            R = R+1./X(I)
  120    CONTINUE
         R = R-1
         TIME2 = DTIME(TARRAY)
         PRINT *,'METHOD 7 GIVES',R
```

```
      TOTALTIME = TOTALTIME+TIME2
      PRINT 40,TIME2,TARRAY(1),TARRAY(2)
130 CONTINUE
    PRINT 140,TOTALTIME
140 FORMAT ('TOTAL TIME ',F8.4,' SECONDS')
    STOP
    END
```

```
C
C     PROBLEM 5
C
C   REFERNCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (NTDIM=50,NSDIM=1000,KASES=5)
      DIMENSION SCORES(NTDIM,NSDIM),TOP(NSDIM),M(KASES)
      DIMENSION N(KASES),TAVER(KASES)
      DIMENSION TLOWABO(KASES)
      LOGICAL ABOVE(NTDIM,NSDIM),GENIUS,TGENIUS(KASES),G(NSDIM)
      REAL LOWABO
      DIMENSION TARRAY(2)
      DATA N / 5,20,30,40,50 /,M / 12,30,200,400,1000 /
      TIME1 = DTIME(TARRAY)
      DO 10 I = 1, NTDIM
         DO 10 J = 1, NSDIM
            SCORES(I,J) = 60.+40.*SIN(I*J*63.21)
   10 CONTINUE
      DO 100 K = 1, KASES
         NABOVE = 0
         LOWABO = 9.9E10
         NT = N(K)
         NS = M(K)
         DO 20 I = 1, NT
            TOP(I) = XMAX(SCORES,NTDIM,NSDIM,I)
   20    CONTINUE
         SUM = 0.0
         DO 30 I = 1, NT
            DO 30 J = 1, NS
               SUM = SUM+SCORES(I,J)
   30    CONTINUE
         AVER = SUM/(NS*NT)
         DO 50 I = 1, NT
            DO 40 J = 1, NS
C
C                    INCREASE SCORES ABOVE AVERAGE
C
               IF (SCORES(I,J).GT.AVER) THEN
                  SCORES(I,J) = 1.1*SCORES(I,J)
               ENDIF
C
C                    SET SWITCH FOR SCORES ABOVE AVERAGE
C
               IF (SCORES(I,J).GT.AVER) THEN
                  NABOVE = NABOVE+1
                  ABOVE(I,J) = .TRUE.
               ENDIF
   40       CONTINUE
   50    CONTINUE
C
C        LOWABO = MASKMN(SCORES, ABOVE)
C
         DO 60 I = 1, NT
            DO 60 J = 1, NS
               IF (ABOVE(I,J)) THEN
                  IF (SCORES(I,J).LT.LOWABO) THEN
                     LOWABO = SCORES(I,J)
                  ENDIF
               ENDIF
   60    CONTINUE
C
C                    FIND THE GENIUSES
C           G(I) = ALL(ABOVE(*,I))
C
         DO 80 I = 1, NT
            G(I) = .TRUE.
            DO 70 J = 1, NS
               IF (.NOT.(ABOVE(I,J))) THEN
                  G(I) = .FALSE.
               ENDIF
   70       CONTINUE
   80    CONTINUE
C
C        GENIUS = ANY(G)
C
         GENIUS = .FALSE.
         DO 90 I = 1, NT
            IF (G(I)) THEN
               GENIUS = .TRUE.
            ENDIF
   90    CONTINUE
         TAVER(K) = AVER
         TGENIUS(K) = GENIUS
         TLOWABO(K) = LOWABO
  100 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 110 K = 1, KASES
         PRINT 105, N(K), M(K)
  105    FORMAT('PROBLEM 5 WITH NT, NS =', I6,2X,I6)
         PRINT *,'GIVES AVERAGE, GENIUS =',TAVER(K),TGENIUS(K)
         PRINT *,'      AND LOW-ABOVE    =',TLOWABO(K)

C
C        WRITE(6,30) (I,TOP(I), G(I), I=1, NT)
C   30   FORMAT(5(I4,F9.4,L2))
C
  110 CONTINUE
      PRINT 120,TIME2,TARRAY(1),TARRAY(2)
  120 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END

      FUNCTION XMAX (A,N,M,I)
C
C           MAX FUNCTION TO HANDLE FINDING THE MAXIMUM
C           VALUE IN A ROW OF AN ARRAY
C
      DIMENSION A(N,M)
      XXMAX = 1.0E-10
      DO 10 JJ = 1, M
         IF (A(I,JJ).GT.XXMAX) THEN
            XXMAX = A(I,JJ)
         ENDIF
   10 CONTINUE
      XMAX = XXMAX
      RETURN
      END
```

```
      PROBLEM 6

REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
            CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
            JOHN R. RICE, MAY 1, 1985
            Corrected Summer of 1989

            REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990

      PARAMETER (MAX=16384,KASES=4)
      REAL L(MAX),D(MAX),U(MAX),X(MAX),Y(MAX),T(MAX)
      INTEGER NI(KASES)
      REAL TARRAY(2),SOLUT(KASES)
      DATA NI / 128,1024,4096,16384 /

      CREATE MATRIX FOR ALL CASES

      NMAX = NI(KASES)
      TIME1 = DTIME(TARRAY)
      DO 10 I = 1, NMAX
         L(I) = .08-.1*SIN(I*12.36)
         D(I) = 1.+.01*COS(I*8.11)
         U(I) = .75+.2*SIN(I*36.12+3.2)
         Y(I) = 1.0
   10 CONTINUE

                              LOOP OVER CASES

      DO 140 M = 1, KASES
         N = NI(M)

         LIMIT = LOG BASE 2 OF N

         LIMIT = 1.44269504*ALOG(FLOAT(N))+.01
         K = 1

                                 MAIN LOOP

         DO 110 II = 1, LIMIT
            DO 20 I = 1, N
               L(I) = L(I)/D(I)
               U(I) = U(I)/D(I)
               Y(I) = Y(I)/D(I)
   20       CONTINUE

                       T IS A TEMPORARY ARRAY
                  COMPUTE AND ASSIGN TO D, COMPUTE Y

            DO 30 I = 1, K
               D(I) = 1.-U(I)*L(I+K)
               T(I) = Y(I)-U(I)*L(I+K)
   30       CONTINUE
            DO 40 I = K+1, N-K
               D(I) = 1.-L(I)*U(I-K)-U(I)*L(I+K)
               T(I) = Y(I)-L(I)*Y(I-K)-U(I)*L(I+K)
   40       CONTINUE
            DO 50 I = N-K+1, N
               D(I) = 1.-L(I)*U(I-K)
               T(I) = Y(I)-L(I)*Y(I-K)
   50       CONTINUE

                          ASSIGN TO Y, COMPUTE L

            DO 60 I = 1, K
               Y(I) = T(I)
               T(I) = 0.
   60       CONTINUE
            DO 70 I = K+1, N
               Y(I) = T(I)
               T(I) = -L(I)*L(I-K)
   70       CONTINUE
C
C                             ASSIGN TO L, COMPUTE U
C
            DO 80 I = 1, N-K
               L(I) = T(I)
               T(I) = U(I)*U(I+K)
   80       CONTINUE
            DO 90 I = N-K+1, N
               L(I) = T(I)
               T(I) = 0.
   90       CONTINUE
C
C                             ASSIGN TO U
C
            DO 100 I = 1, N
               U(I) = T(I)
  100       CONTINUE
            K = 2*K
  110    CONTINUE
         DO 120 I = 1, N
            X(I) = Y(I)/D(I)
  120    CONTINUE
         SUMX = 0
         DO 130 I = 1, N
            SUMX = SUMX+X(I)
  130    CONTINUE
         SOLUT(M) = SUMX
  140 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 150 M = 1, KASES
         PRINT *,'PROBLEM 6   WITH N =',NI(M)
         PRINT *,'GIVES SOLUTION =',SOLUT(M)
  150 CONTINUE
      PRINT 160,TIME2,TARRAY(1),TARRAY(2)
  160 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END
```

```
C
C     PROBLEM 7
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (MAXPT=50,KASES=4)
      DIMENSION XI(MAXPT),XL(MAXPT)
      DIMENSION TEMP(MAXPT),TAMP(MAXPT),TUMP(MAXPT)
      DIMENSION X(5),P(KASES,5),TP(5),NK(KASES)
      DOUBLE PRECISION DENOM(MAXPT)
      DIMENSION TARRAY(2)
      DATA X / 1.1,1.2,2.1,-1.1,2.2 /,NK / 5,20,80,320 /

      TIME1 = DTIME(TARRAY)
      DO 90 NN = 1, KASES
         N = NK(NN)
C
C        FORALL(I=1:N) XI(I) = I*.08
C
         DO 10 I = 1, N
            XI(I) = I*.08
   10    CONTINUE
C
C              DENOMENATOR
C
         DO 30 I = 1, N
            TEMP(I) = 1.
            DENOM(I) = 1.0
            DO 20 J = 1, N
               IF (J.NE.I) THEN
                  TEMP(J) = XI(I)-XI(J)
                  DENOM(I) = DENOM(I)*TEMP(J)
               ENDIF
   20       CONTINUE
C
C       NOTE: THE DENOMINATOR IS INVERTED HERE SO THAT A MULTIPLICATION
C              CAN BE DONE LATER
C
            DENOM(I) = 1.0/DENOM(I)
   30    CONTINUE
         DO 80 K = 1, 5
            DO 60 I = 1, N
               DO 40 J = 1, N
                  IF (J.NE.I) THEN
                     TAMP(J) = X(K)-XI(J)
                  ENDIF
   40          CONTINUE
               TAMP(I) = 1.0
               PTAMP = 1.0
               DO 50 J = 1, N
                  PTAMP = PTAMP*TAMP(J)
   50          CONTINUE
               XL(I) = PTAMP*DENOM(I)
   60       CONTINUE
C
C         FORALL (I=1:N) TUMP(I)   F(XI(I)) * XL(I)
C
            P(NN,K) = 0
            DO 70 I = 1, N
               TUMP(I) = F(XI(I))*XL(I)
               P(NN,K) = P(NN,K)+TUMP(I)
   70       CONTINUE
   80    CONTINUE
   90 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 120 NN = 1, KASES
         DO 100 I = 1, 5
            TP(I) = P(NN,I)
  100    CONTINUE
         PRINT *,'PROBLEM 7 WITH N =',NK(NN)
         WRITE (6,110) TP
  110    FORMAT ('GIVES P(X) VALUES ='/6(1X,E14.7))
  120 CONTINUE
      PRINT 130,TIME2,TARRAY(1),TARRAY(2)
  130 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END

      FUNCTION F (X)
      F = EXP(X)
      RETURN
      END
```

```
BLEM 8

NCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
      C6D-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
      JOHN R. RICE, MAY 1, 1985

      REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990

AMETER (NDIM=3200,MDIM=40,KASES=5)
ENSION DIFF(NDIM,MDIM),X(NDIM),TSUMD(KASES)
EGER NPT(KASES),MDIFF(KASES),TN(KASES),TM(KASES)
:NSION TARRAY(2)
A NPT / 6,50,200,800,3200 /,MDIFF / 3,6,12,20,40 /
31 = DTIME(TARRAY)
10 NK = 1, KASES
 = NPT(NK)
 = MDIFF(NK)

ORALL(I=1:N) X(I)= .2*I + .01 * COS(I)

O 10 I = 1, N
   X(I) = .2*I+.01*COS(FLOAT(I))
   DIFF(I,1) = SIN(X(I))
CONTINUE
SUMD = 0
O 30 J = 2, M
   N = N-1
   DO 20 I = 1, N
      DIFF(I,J) = (DIFF(I+1,J-1)-DIFF(1,J-1))/(X(I+J-1)-X(I))
      SUMD = SUMD+DIFF(I,J)
   CONTINUE
CONTINUE
N(NK) = N
M(NK) = M
SUMD(NK) = SUMD
 INUE
2 = DTIME(TARRAY)
50 NK = 1, KASES
PRINT 50,TN(NK),TM(NK)
ORMAT ('PROBLEM 8 WITH N,M =',I6,2X,I6)

JMD = SUM(DIFF(*,1:M))

PRINT *,'GIVES SUM OF DIFFERENCE TABLE = ',TSUMD(NK)
TINUE
NT 70,TIME2,TARRAY(1),TARRAY(2)
MAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
P
```

```
C
C     PROBLEM 9
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (NDIM=200,MDIM=200,KASES=4)
      DIMENSION U(NDIM,MDIM),T(NDIM,MDIM),TSUMT(KASES)
      DIMENSION TARRAY(2)
      INTEGER NK(KASES),MK(KASES)
      DATA NK / 4,10,80,200 /,MK / 4,20,60,180 /
C
      TIME1 = DTIME(TARRAY)
      DO 80 KK = 1, KASES
         N = NK(KK)
         M = MK(KK)
C
C        FORALL (I=1:N, J=1:M) U(I,J) = I*(I+1) + J/(J+1)
C
         DO 20 I = 1, N
            DO 10 J = 1, M
               U(I,J) = I*(I+1)+J/(J+1)
   10       CONTINUE
   20    CONTINUE
         DO 40 I = 2, N-1
            DO 30 J = 2, M-1
               T(I,J) = (U(I,J)+U(I+1,J)+U(I-1,J)+U(I,J+1)+U(I+1,J+1)+U(
     *            I-1,J+1)+U(I,J-1)+U(I+1,J-1)+U(I-1,J-1))/9.
   30       CONTINUE
            T(I,1) = (U(I,1)+U(I,2)+U(I-1,1)+U(I+1,1)+U(I-1,2)+U(I+1,2))
     *         /6.
            T(I,M) = (U(I,M)+U(I,M-1)+U(I-1,M)+U(I+1,M)+U(I-1,M-1)+U(I+1
     *         ,M-1))/6.
   40    CONTINUE
         DO 50 J = 1, M-1
            T(1,J) = (U(1,J)+U(1,J+1)+U(1,J-1)+U(2,J)+U(2,J+1)+U(2,J-1))
     *         /6.
            T(N,J) = (U(N,J)+U(N,J+1)+U(N,J-1)+U(N-1,J)+U(N-1,J+1)+U(N-1
     *         ,J-1))/6.
   50    CONTINUE
         T(1,1) = (U(1,1)+U(1,2)+U(2,2)+U(2,1))/4.
         T(N,1) = (U(N,1)+U(N,2)+U(N-1,2)+U(N-1,N-1))/4.
         T(1,M) = (U(1,M)+U(2,M)+U(1,M-1)+U(2,M-1))/4.
         T(N,M) = (U(N,M)+U(N-1,M)+U(N,M-1)+U(N-1,M-1))/4.
C
C        SUMT = SUM(T)
C        PRODT = PRODUCT(T)
C
         SUMT = 0.0
         DO 70 I = 1, N
            DO 60 J = 1, M
               SUMT = SUMT+T(I,J)
   60       CONTINUE
   70    CONTINUE
         TSUMT(KK) = SUMT
   80 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 90 KK = 1, KASES
         PRINT 100,NK(KK),MK(KK)
         PRINT*,'GIVES SUM  = ',TSUMT(KK)
   90 CONTINUE
      
  100 FORMAT ('PROBLEM 9 WITH N,M =',I6,2X,I6)
      PRINT 110,TIME2,TARRAY(1),TARRAY(2)
  110 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      PRINT *,
     *   '(PRODUCT CALCULATION CAUSED ARITHMETIC OVERFLOW ON THE   VAX)'
      STOP
      END
```

```
C
C     PROBLEM 10
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (NDIM=100,KASES=5)
      DIMENSION A(NDIM,NDIM),TEMP(NDIM),NK(KASES)
      DIMENSION TSUMD(KASES),TSUMA(KASES)
      DIMENSION TARRAY(2)
      DATA NK / 4,10,20,40,100 /
      TIME1 = DTIME(TARRAY)
C
C           LOOP OVER 5 MATRIX SIZES
C
      DO 80 NN = 1, KASES
         N = NK(NN)
C
C        FORALL (I=1:N, J=1:N) A(I,J) = SIN(I+J)
C
         DO 10 I = 1, N
            DO 10 J = 1, N
               A(I,J) = SIN(FLOAT(I+J))
   10    CONTINUE
C
C        LOOP OVER COLUMNS
C
         DO 50 ICOL = 1, N
            COLMAX = XMAX(A,N,NDIM,ICOL,IMAX)
C
C           IMAX   = LOCMAX(A(ICOL:N,ICOL))
C                  INTERCRANGE ROWS
C
            DO 20 J = ICOL, N
               TEMP(J) = A(IMAX,J)
               A(IMAX,J) = A(ICOL,J)
               A(ICOL,J) = TEMP(J)
   20       CONTINUE
            DO 40 JROW = ICOL+1, N
               A(ICOL,JROW) = A(ICOL,JROW)/COLMAX
               DO 30 K = ICOL+1, N
                  A(JROW,K) = A(JROW,K)-A(ICOL,K)*A(JROW,ICOL)
   30          CONTINUE
   40       CONTINUE
   50    CONTINUE
C
C        SUMA = SUM(A)
C
         SUMA = 0
         SUMD = 0
         DO 70 I = 1, N
            DO 60 J = 1, N
               SUMA = SUMA+A(I,J)
               IF (I.EQ.J) THEN
                  SUMD = SUMD+A(I,J)
               ENDIF
   60       CONTINUE
   70    CONTINUE
         TSUMA(NN) = SUMA
         TSUMD(NN) = SUMD
   80 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 100 NN = 1, KASES
         PRINT *,'PROBLEM 10 WITH N =',NK(NN)
         SUMA = TSUMA(NN)
         SUMD = TSUMD(NN)
         PRINT 90,SUMA,SUMD
   90    FORMAT ('SUM OF ELEMENTS AND SUM OF DIAGONAL =',2E15.6)
  100 CONTINUE
      PRINT 110,TIME2,TARRAY(1),TARRAY(2)
  110 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END

      FUNCTION XMAX (X,N,NDIM,ICOL,INDEX)
      DIMENSION X(NDIM,NDIM)
      XXMAX = -1.0E-10
      DO 10 J = ICOL+1, N
         IF (ABS(X(J,ICOL)).GT.XXMAX) THEN
            XXMAX = X(J,ICOL)
            INDEX = J
         ENDIF
   10 CONTINUE
      XMAX = XXMAX
      RETURN
      END
```

```
C
C     PROBLEM 11
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (NMAX=5000,KASES=4)
      DIMENSION DATA(NMAX),COSWT(NMAX),NK(KASES),FMOM(4)
      DIMENSION TARRAY(2),TFMOM(KASES,4)
      DATA PI / 3.1415926 /,NK / 6,200,1000,5000 /
      TIME1 = DTIME(TARRAY)
      DO 60 KK = 1, KASES
         N = NK(KK)
C
C          FORALL (I=1:N) DATA(I) = -I/0.1 + 1000 * SIN(I+10.)
C
         DO 10 I = 1, N
            DATA(I) = -I/0.1+1000*SIN(I+10.)
   10    CONTINUE
C
C                         TRIM DATA
C
         DO 20 I = 1, N
            DATA(I) = AMAX1(0.0,AMIN1(1000.,DATA(I)))
   20    CONTINUE
C
C          FORALL(I=1:N) DATA(I) = LOG(DATA(I) + 1.0)
C
         DO 30 I = 1, N
            DATA(I) = ALOG(DATA(I)+1.0)
   30    CONTINUE
         TFMOM(KK,1) = SUM(DATA,N)/N
         DO 50 K = 2, 4
            DO 40 I = 1, N
               COSWT(I) = COS(PI*I*(K-1)/(N+1)) * DATA(I)
   40       CONTINUE
            TFMOM(KK,K) = SUM(COSWT,N)/N
   50    CONTINUE
   60 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 90 KK = 1, KASES
         PRINT *,'PROBLEM 11 WITH N =',NK(KK)
         DO 70 I = 1, 4
            FMOM(I) = TFMOM(KK,I)
   70    CONTINUE
         PRINT 80,KK,FMOM(1),FMOM(2),FMOM(3),FMOM(4)
   80    FORMAT (I6,F8.4,F8.4,F8.4,F8.4)
   90 CONTINUE
      PRINT 100,TIME2,TARRAY(1),TARRAY(2)
  100 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END
C
C      SUBROUTINE SAVMOM(I,FVAL,N)
C      DIMENSION FVAL(N)
C      PRINT *, I, FVAL
C      RETURN
C      END
C
      FUNCTION SUM (X,N)
      DIMENSION X(N)
      S = 0
      DO 10 I = 1, N
         S = S+X(I)
   10 CONTINUE
      SUM = S
      RETURN
      END
```

```
C
C     PROBLEM 12
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (ND=400,MD=400,ND1=ND+1,MD1=MD+1,KASES=5)
      DIMENSION A(MD,ND),R(MD),C(ND),ABIG(MD1,ND1)
      DIMENSION TARRAY(2),TP1(KASES),TP2(KASES)
      INTEGER NK(KASES),MK(KASES)
      DATA NK / 4,10,50,100,400 /,MK / 4,8,60,100,400 /

      TIME1 = DTIME(TARRAY)
      DO 70 K = 1, KASES
         N = NK(K)
         M = MK(K)
C
C           FORALL(I=1:M) R(I) = 1. - I
C           FORALL(J=1:N) C(J) = 1. + J
C           FORALL(I=1:M, J=1:N) A(I,J) = I+J
C
         DO 20 I = 1, M
            R(I) = 1.-I
            DO 10 J = 1, N
               A(I,J) = I+J
   10       CONTINUE
   20    CONTINUE
         DO 30 J = 1, N
            C(J) = 1.+J
   30    CONTINUE
         ACORN = .5
C
C                 BUILD ABIG FROM PARTS CREATED
C           FORALL(I=1:M, J=1,N)   ABIG(I,J) = A(I,J)
C           FORALL(I=1,N)          ABIG(I,N+1) = C(I)
C           FORALL(J=1,M)          ABIG(M+1,J) = R(J)
C
         DO 50 I = 1, M
            ABIG(I,N+1) = R(I)
            DO 40 J = 1, N
               ABIG(I,J) = A(I,J)
   40       CONTINUE
   50    CONTINUE
         DO 60 J = 1, N
            ABIG(M+1,J) = C(J)
   60    CONTINUE
         ABIG(M+1,N+1) = ACORN
         TP1(K) = ABIG(1,1)*ABIG(1,N+1)*ABIG(M+1,N)*ABIG(M+1,N+1)
         TP2(K) = ABIG(M,N)*ABIG(M,N+1)*ABIG(M+1,N)*ABIG(M+1,N+1)
   70 CONTINUE
      TIME2 = DTIME(TARRAY)
      DO 90 K = 1, KASES
         PRINT 80,NK(K),MK(K)
   80    FORMAT ('PROBLEM 12 WITH N,M =',I6,2X,I6)
         PRINT*,'GIVES CORNER PRODUCTS =', TP1(K),TP2(K)
   90 CONTINUE
      PRINT 100,TIME2,TARRAY(1),TARRAY(2)
  100 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END
```

```
      IF (SIN(A(I)).GE.
        A(I) = A(I)+C(
      ELSE
        A(I) = A(I)-D(
      ENDIF
      E(K) = E(K)+A(I)*
   20 CONTINUE
   30 CONTINUE
      TIME2 = DTIME(TARRAY
      DO 40, K = 1, KASES
      PRINT *,'PROBLEM 13 I
      PRINT *,'GIVES E =',E
   40 CONTINUE
      PRINT 50,TIME2,TARRA
   50 FORMAT ('TIME ',F8.4
      STOP
      END
```

TEST PARALLEL AND VECTOR LANGUAGES
COMPUTER SCIENCE, PURDUE UNIVERSITY
, MAY 1, 1985

JOHN R. RICE AND J. JING, OCT. 1, 1990

```
),KASES*3)
NDIM),C(NDIM),D(NDIM)
3(KASES),N(KASES)
, 21000 /


/10. + 1./I

)


)+.02
)*SIN(A(I))
?.*C(I)

CREATED THE 4 VECTORS OF DATA


9(I))
COS(C(I))) THEN
I)

I)

*2

FOR N =',N(K)
3(K)

(1),TARRAY(2)
 SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
```

```
C
C     PROBLEM 13
C
C  REFERENCE:  PROBLEMS TO
C              CSD-TR 516,
C              JOHN R. RIC
C
C              REVISED BY
C
      PARAMETER (NDIM=2100
      DIMENSION A(NDIM),B(
      DIMENSION TARRAY(2),
      DATA N / 1000, 10000
C
C     CRITICAL E
C
C     FORALL(I=*) A(I) = I,
C
      TIME1 = DTIME(TARRAY
      DO 30 K = 1, KASES
      DO 10 I = 1, N(K)
         A(I) = I/10.+1./I
         B(I) = LOG10(A(I)
         C(I) = (A(I)+B(I)
         D(I) = A(I)+D(I)-
   10 CONTINUE
C
C                    HAVE
C
      E(K) = 0.0
      DO 20 I = 1, N(K)
```

```
C
C     PROBLEM 14
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
C     PAGE 204 OF NUMERICAL METHODS, SOFTWARE AND ANALYSIS
C
      PARAMETER (NMETH=3,NFUNK=3,NACCUR=11)
      REAL SUM1(NACCUR),SUM21(NACCUR),SUM22(NACCUR),SUM31(NACCUR),SUM32(
     *   NACCUR),SUM33(NACCUR)
      INTEGER NSIMP(NACCUR)
      REAL H(NACCUR),HS(NACCUR),HG(NACCUR),H77(NACCUR)
      REAL RESULT(NFUNK,NACCUR,NMETH,2)
      REAL TARRAY(2),TIME1,TIME2

      INTEGER N(NACCUR)
      COMMON /SELECT/ JFUNK
      DATA N / 10,25,50,75,100,150,200,300,500,1000,2000 /

      PRINT *,'PROBLEM 14'
      TIME1 = DTIME(TARRAY)
      DO 100 JFUNK = 1, NFUNK
C
C        DO PARALLEL
C           BEGIN       'METHOD 1'
C
         CALL FVALS (A1,B1,TRUE,JFUNK)
         DO 20 L = 1, NACCUR
C
C              BEGIN
C
            H(L) = (B1-A1)/N(L)
C
C              SUM1(L) = SUM(F(A1+H(L)*SEQ(1,N(L)-1)))
C
            SUM1(L) = 0
            DO 10 JJ = 1, N(L)-1
               SUM1(L) = SUM1(L)+F(A1+H(L)*JJ)
   10       CONTINUE
            SUM1(L) = (SUM1(L)*2.+F(A1)+F(B1))*H(L)/2.
            RESULT(JFUNK,L,1,2) = N(L)+1
            RESULT(JFUNK,L,1,1) = SUM1(L)
C
C              END
C
   20    CONTINUE
C
C           END       'METHOD 1'
C           BEGIN     'METHOD 2'
C
         CALL FVALS (A2,B2,TRUE,JFUNK)
         DO 50 L = 1, NACCUR
C
C              BEGIN
C
            NSIMP(L) = N(L)
            IF (MOD(NSIMP(L),2).EQ.1) NSIMP(L) = NSIMP(L)-1
            HS(L) = (B2-A2)/NSIMP(L)
C
C              SUM21(L) = SUM(F(A2 + HS(L)*SEQ(1,NSIMP(L)-1,2) ))
C
            SUM21(L) = 0
            DO 30 JJ = 1, NSIMP(L)-1, 2
               SUM21(L) = SUM21(L)+F(A2+HS(L)*JJ)
   30       CONTINUE
C
C              SUM22(L) = SUM(F(A2 + HS(L)*SEQ(2,NSIMP(L)-2,2)))
C
            SUM22(L) = 0
            DO 40 JJ = 2, NSIMP(L)-2, 2
               SUM22(L) = SUM22(L)+F(A2+HS(L)*JJ)
   40       CONTINUE
            RESULT(JFUNK,L,2,1) = HS(L)*(F(A2)+F(B2)+4.*SUM21(L)+2.*
     *         SUM22(L))/3.
            RESULT(JFUNK,L,2,2) = NSIMP(L)+1
C
C              END
C
   50    CONTINUE
C
C           END       'METHOD 2'
C           BEGIN     'METHOD 3'
C
         CALL FVALS (A3,B3,TRUE,JFUNK)
         DO 90 L = 1, NACCUR
C
C              BEGIN
C
            HG(L) = (B3-A3)/INT(N(L)/3)
            H77(L) = .774596669241*HG(L)/2.
C
C              SUM31(L) = SUM(F(A3+H77(L)+HG(L)*SEQ(1,N(L)/3)/2.))
C
            SUM31(L) = 0
            DO 60 JJ = 1, N(L)/3
               SUM31(L) = SUM31(L)+F(A3+H77(L)+HG(L)*JJ/2.)
   60       CONTINUE
C
C              SUM32(L) = SUM(F(A3+HG(L)*SEQ(1,N(L)/3)/2.))
C
            SUM32(L) = 0
            DO 70 JJ = 1, N(L)/3
               SUM32(L) = SUM32(L)+F(A3+HG(L)*JJ/2.)
   70       CONTINUE
C
C              SUM33(L) = SUM(F(A3+H77(L)+HG(L)*SEQ(1,(N(L)/3)/2.)))
C
            SUM33(L) = 0
            DO 80 JJ = 1, N(L)/3
               SUM33(L) = SUM33(L)+F(A3+H77(L)+HG(L)*JJ/2.)
   80       CONTINUE
            RESULT(JFUNK,L,3,1) = HG(L)*(5.*(SUM31(L)+SUM33(L))+8*SUM32(
     *         L))/18.
            RESULT(JFUNK,L,3,2) = 3.*INT(N(L)/3)
C
C              END
C
   90    CONTINUE
C
C           END       'METHOD 3'
C
  100 CONTINUE
      TIME2 = DTIME(TARRAY)
C
```

```
C     ENDALL
C
      DO 160 JFUNK = 1, NFUNK
         CALL FVALS (A,B,TRUE,JFUNK)
         PRINT *,' '
         PRINT 110,JFUNK,A,B,TRUE
  110    FORMAT (' FUNCTION',I3,2X,'ON',2X,2F5.2,2X,'HAS TRUE =',F10.6)
         DO 150 L = 1, NACCUR, NACCUR-1
            PRINT *,' EVALUATION WITH',N(L),' INTERVALS, L =',L
            PRINT 120
  120       FORMAT (2X,'METHOD',8X,'ANSWER',5X,'NO OF POINTS',5X,'ERROR'
     *         ,5X,'LOG ERROR')
            DO 140 NL = 1, NMETH
               PRINT 130,NL,RESULT(JFUNK,L,NL,1),RESULT(JFUNK,L,NL,2),
     *            TRUE-RESULT(JFUNK,L,NL,1),ALOG10(AMAX1(ABS(TRUE-RESULT
     *            (JFUNK,L,NL,1)),.1E-30))
  130          FORMAT (I6,2X,F16.12,2X,F6.0,2X,F16.12,2X,E18.12)
  140       CONTINUE
  150    CONTINUE
  160 CONTINUE
      PRINT 170,TIME2,TARRAY(1),TARRAY(2)
  170 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END

      FUNCTION F (X)
      COMMON /SELECT/ JFUNK
      IF (JFUNK.EQ.1) F = EXP(X)
      IF (JFUNK.EQ.2) F = SQRT(ABS(X-.2345))
      IF (JFUNK.EQ.3) F = 1.+X*X+1./(1.+100.*X*X)
      RETURN
      END
      SUBROUTINE FVALS (A,B,TRUE,JFUNK)
      IF (JFUNK.EQ.1) THEN
         A = 0.
         B = 1.
         TRUE = 1.71828182845
      ENDIF
      IF (JFUNK.EQ.2) THEN
         A = 0.
         B = 1.
         TRUE = .5222099422093
      ENDIF
      IF (JFUNK.EQ.3) THEN
         A = -1.
         B = 2.
         TRUE = 6.29919656054
      ENDIF
      RETURN
      END
```

```
C
C     PROBLEM 15
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (NPTS=10,NMAX=2*NPTS,KNOTSMAX=NPTS+2,N20=20*NPTS)
      COMMON /POINTS/ N,KASE
      REAL XPTS(NPTS,NPTS,2),COEF(NMAX,NPTS,2),H(NPTS)
      REAL ERRMAX(NPTS,2),TKNOTS(KNOTSMAX),TARRAY(2),DECAY(KNOTSMAX,2)
      REAL LOGERR,INTERP,RATIO,TIME1,TIME2
      DATA A,B / -1.,1. /,PI / 3.141592654 /
      TIME1 = DTIME(TARRAY)
      DO 100 KASE = 1, 2
C
C        BEGIN              'NUMBER OF POINTS USED'
C
         DO 80 N = 2, NPTS
C
C           BEGIN                 'CASES OF POINT DISTRIBUTION'
C
            IF (KASE.EQ.1) THEN
               H(N) = (B-A)/(N-1)
               DO 10 J = 1, N
                  XPTS(J,N,KASE) = A+(J-1)*H(N)
   10          CONTINUE
            ELSE
               DO 20 J = 1, N
                  XPTS(J,N,KASE) = (A+B)/2+(A-B)/2*COS((2*J-1)*PI/(2*N))
   20          CONTINUE
               RATIO = (B-A)/(XPTS(N,N,KASE)-XPTS(1,N,KASE))
               DO 30 J = 1, N
                  XPTS(J,N,KASE) = (XPTS(J,N,KASE)-(A+B)/2)*
     .                             RATIO+(A+B)/2
   30          CONTINUE
            ENDIF
C
C           DO PARALLEL OVER J,N,KASE: 'HERMITE CUBIC COEFFICIENTS'
C              FORALL(KASE=1,2;N=2,NPTS;J=1,2N,2)
C                                 COEF(J,N,KASE) = F(XPTS(J,N,KASE))
C              FORALL(KASE=1,2;N=2,NPTS;J=2,2N,2)
C                                 COEF(J,N,KASE) = FPRIME(XPTS(J,N,KASE)
C
            DO 40 J = 1, N
               COEF(2*J-1,N,KASE) = F(XPTS(J,N,KASE))
   40       CONTINUE
            DO 50 J = 1, N
               COEF(2*J,N,KASE) = FPRIME(XPTS(J,N,KASE))
   50       CONTINUE
C
C           END PARALLEL OVER J,N,KASE
C
C           DO PARALLEL OVER N,KASE: 'EVALUATE ERRORS OF INTERPOLATION'
C           PUT XPTS POINTS INTO TKNOTS ARRAY OF HERMITE CUBICS KNOTS
C
            DO 60 J = 1, N
               TKNOTS(J+1) = XPTS(J,N,KASE)
   60       CONTINUE
C
C           ADD THE DUMMY POINTS AT EACH END OF TKNOTS ARRAY
C
            TKNOTS(1) = A-0.1*(ABS(A)+1.)
            TKNOTS(N+2) = B+0.1*(ABS(B)+1.)
            KNOTS = N+2
            ERRMAX(N,KASE) = -10E20
            DO 70 I = 1, N20
               X = A+(I-1)*(B-A)/(N20-1)
               ERROR = ABS(F(X)-INTERP(X,COEF,TKNOTS,KNOTS,NPTS,NMAX,
     .                 KNOTSMAX))
               IF (ERROR.GT.ERRMAX(N,KASE)) THEN
                  ERRMAX(N,KASE) = ERROR
               ENDIF
   70       CONTINUE
C
C           END                'NUMBER OF POINTS USED'
C
   80    CONTINUE
         DO 90 N = 3, NPTS
            OLDERR = ALOG(ERRMAX(N-1,KASE))
            LOGERR = ALOG(ERRMAX(N,KASE))
            DECAY(N,KASE) = (LOGERR-OLDERR)/ALOG(FLOAT(N)/FLOAT(N-1))
   90    CONTINUE
C
C           END PARALLEL OVER N,KASE
C
C        END                   'CASES OF POINT DISTRIBUTION'
C
  100 CONTINUE
      TIME2 = DTIME(TARRAY)-TIME1
C
C             COMPUTATION OUTPUT IS SEQUENTIAL
C
      DO 170 KASE = 1, 2
         IF (KASE.EQ.1) THEN
            PRINT 110
  110       FORMAT (4X,'PROBLEM 15'/)
            PRINT 120
  120       FORMAT (9X,'EQUALLY SPACED POINTS'/)
            PRINT 130
  130       FORMAT (4X,'N      MAX ERROR      DECAY EXPONENT')
         ELSE
            PRINT 140
  140       FORMAT (/9X,'CHEBYSHEV SPACED POINTS'/)
            PRINT 130
         ENDIF
         DO 160 N = 2, NPTS
            PRINT 150,N,ERRMAX(N,KASE),DECAY(N,KASE)
  150       FORMAT (1X,I4,4X,E12.4,3X,F11.2)
  160    CONTINUE
C
C           PRINT OUT OF INTERPOLATION POINT ARRAY XPTS(J,N,KASE)
C           SUPPRESSED HERE TO REDUCE AMOUNT OF OUTPUT
C
  170 CONTINUE
      PRINT 180,TIME2,TARRAY(1),TARRAY(2)
  180 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END

      REAL FUNCTION INTERP (X,COEF,T,KNOTS,NPTS,NMAX,KNOTSMAX)
      REAL COEF(NMAX,NPTS,2),T(KNOTSMAX)
      COMMON /POINTS/ N,KASE
C
C        PUT THE APPROPRIATE ARRAY OF KNOTS INTO THE T ARRAY
C
```

```
C     FORALL(J=1,2*N) V(J) = COEF(J)*HCUBIC(J,X)
C     INTERP = SUM(V(1:2*N))
C
      INTERP = 0
      DO 10 J = 1, 2*N
         INTERP = INTERP+COEF(J,N,KASE)*HCUBIC(J,X,T,KNOTS)
   10 CONTINUE
      IF (INTERP.EQ.0) THEN
         INTERP = F(X)
      ENDIF
      RETURN
      END

      REAL FUNCTION HCUBIC (J,X,T,KNOTS)
C
C         THE HERMITE CUBIC KNOTS ARE IN THE ARRAY T OF LENGTH KNOTS
C           THERE ARE 2 SUCH ARRAYS, 1 FOR EACH CASE OF POINT DIST.
C         J = INDEX OF THE BASIS FUNCTION
C         X = POINT OF EVALUATION OF THE BASIS FUNCTION
C
      REAL T(KNOTS)
      IKNOT = (J+3)/2
      IF (MOD(J,2).EQ.1) THEN
         HCUBIC = HERMC(T,KNOTS,X,IKNOT)
      ELSE
         HCUBIC = HERMC1(T,KNOTS,X,IKNOT)
      ENDIF
      RETURN
      END

      REAL FUNCTION HERMC (T,KNOTS,X,IKNOT)
      REAL T(KNOTS)
      IF ((X.LT.T(IKNOT+1)).AND.(X.GT.T(IKNOT-1))) THEN
         HERMC = 0.0
         IF (X.GT.T(IKNOT)) THEN
            DT = T(IKNOT+1)-T(IKNOT)
            DX = T(IKNOT+1)-X
         ELSE
            DT = T(IKNOT)-T(IKNOT-1)
            DX = X-T(IKNOT-1)
         ENDIF
         HERMC = (3.-2.*DX/DT)*(DX**2)/DT**2
      ELSE
         HERMC = 0.0
      ENDIF
      RETURN
      END

      REAL FUNCTION HERMC1 (T,KNOTS,X,IKNOT)
      REAL T(KNOTS)
      IF ((X.LT.T(IKNOT+1)).AND.(X.GT.T(IKNOT-1))) THEN
         DX = X-T(IKNOT)
         IF (X.GT.T(IKNOT)) THEN
            DT2 = (T(IKNOT)-T(IKNOT+1))**2
            DX2 = (X-T(IKNOT+1))**2
         ELSE
            DT2 = (T(IKNOT)-T(IKNOT-1))**2
            DX2 = (X-T(IKNOT-1))**2
         ENDIF
         HERMC1 = DX2*DX/DT2
      ELSE
         HERMC1 = 0.0
      ENDIF
      RETURN
      END

      REAL FUNCTION F (X)
      F = 1./(X*X+25.0)
      RETURN
      END
      REAL FUNCTION FPRIME (X)
      FPRIME = -2.0*X/(X*X+25.)**2
      RETURN
      END
```

```
C
C     PROBLEM 16
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
C PAGE 152 OF NUMERICAL METHODS, SOFTWARE AND ANALYSIS
C TEST SOLVING HILBERT MATRIX OF SEVERAL SIZES ON 4 RIGHT SIDES
C

      PARAMETER (SIZE=5,KASES=4,NDIM=SIZE*KASES)
      REAL HILBERT,HILB,X,B,TT,TRESID,WORK,HDPROD,DOTPROD
      REAL TARRAY,TIME1,TIME2
      INTEGER I,J,N,IPVT
      DIMENSION HILBERT(NDIM,NDIM),HILB(NDIM,NDIM),X(NDIM,KASES)
      DIMENSION B(NDIM,KASES),WORK(NDIM),IPVT(NDIM)
      DIMENSION TRESID(NDIM,KASES),TT(NDIM)
      DIMENSION XTEMP(NDIM),TARRAY(2)
C
C INITIALIZATION: CREATE HILBERT MATRIX.
C
      TIME1 = DTIME(TARRAY)
      DO 10 I = 1, NDIM
         DO 10 J = 1, NDIM
            HILBERT(I,J) = 1./(I+J-1.0)
   10 CONTINUE
C
C SOLVE THE PROBLEM FOR NO OF SIZES
C
      DO 90 N = KASES, NDIM, KASES
C
C        DEFINE HILBERT MATRIX , AND FIRST 3 RIGHT HAND SIDES
C
         DO 20 I = 1, N
            B(I,1) = 0.0
            B(I,2) = 1.0
            B(I,3) = 1.+.01*SIN(100.*I)
            DO 20 J = 1, N
               HILB(I,J) = HILBERT(I,J)
   20    CONTINUE
         B(1,1) = 1.0
C
C   COMPUTE 4-TH SIDE TO MAKE SOLUTION = 1
C
         DO 30 I = 1, N
            B(I,4) = 0.0
            DO 30 J = 1, N
               B(I,4) = B(I,4)+HILB(I,J)*(-1)**(J+1)
   30    CONTINUE
C
C   ASSIGN X = B
C
         DO 40 I = 1, N
            DO 40 J = 1, KASES
               X(I,J) = B(I,J)
   40    CONTINUE
C
C   USE LINPACK AS FORTRAN 66 ROUTINES
C
         CALL SGECO (HILB,NDIM,N,IPVT,RCOND,WORK)
         IF ((1+RCOND).EQ.1.) THEN
            TT(N) = 1
         ELSE
C
C   BACK SOLVE TO GET FOUR SOLUTIONS
C
            TT(N) = 0
            DO 80 K = 1, KASES
               DO 50 I = 1, N
                  XTEMP(I) = X(I,K)
   50          CONTINUE
               CALL SGESL (HILB,NDIM,N,IPVT,XTEMP,0)
C
C     COMPUTE DOTPRODUCT FOR RESIDUE
C
               DOTPROD = 0.0
               DO 70 I = 1, N
                  HDPROD = 0.0
                  DO 60 J = 1, N
                     HDPROD = HDPROD+HILBERT(I,J)*XTEMP(J)
   60             CONTINUE
                  HDPROD = HDPROD-B(I,K)
                  DOTPROD = DOTPROD+HDPROD*HDPROD
   70          CONTINUE
               TRESID(N,K) = SQRT(DOTPROD)
   80       CONTINUE
         ENDIF
   90 CONTINUE
C
C TIME2 = DTIME(TARRAY)
C
      TIME2 = DTIME(TARRAY)
      DO 120 N = KASES, NDIM, KASES
         IF (TT(N).EQ.1.) THEN
            PRINT *,'PROBLEM 16 WITH N=',N
            PRINT *,'*** THE CONDITION NUMBER IS TOO HIGH'
         ELSE
C
C   PRINT*, 'PROBLEM 16 WITH N = ', N, ' RESIDUE = ', TRESID(N,K)
C
            PRINT *,'PROBLEM 16 WITH N=',N
            DO 100 K = 1, 4
  100          PRINT 110,K,TRESID(N,K)
         ENDIF
  110    FORMAT(2X,'RESIDUE',I4,2X,'=',2X,E18.12)
  120 CONTINUE
      PRINT 130,TIME2,TARRAY(1),TARRAY(2)
  130 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END
```

```
C
C     PROBLEM 17
C
C  REFERENCE:  PROBLEMS TO TEST PARALLEL AND VECTOR LANGUAGES
C              CSD-TR 516, COMPUTER SCIENCE, PURDUE UNIVERSITY
C              JOHN R. RICE, MAY 1, 1985
C
C              REVISED BY JOHN R. RICE AND J. JING, OCT. 1, 1990
C
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      COMMON /PROBLEM/ A,B,EPS
      COMMON /CONTROL/ AREA,BOUNDA,DISCARD,FINISH
      COMMON /QUEUE/ LEADER,NQ,LASTQ,INEXT(LIMQ),AEST(LIMQ),BOUND(LIMQ)
      COMMON /QUEUE/ INFLECT(LIMQ),COTAN(LIMQ),COTR(LIMQ),COTL(LIMQ)
      COMMON /QUEUE/ XRIGHT(LIMQ),FRIGHT(LIMQ),IRIGHT(LIMQ),XLEFT(LIMQ)
      COMMON /QUEUE/ FLEFT(LIMQ),ILEFT(LIMQ),INQUEUE(LIMQ),IQFREE,IDQ
      COMMON /QUEUE/ WAITING,NEXTQ,ITFREE,IDT,TAILING,NEXTT,COUNT
      COMMON /PROCSRS/ ACHANGE(LIMCPU),BCHANGE(LIMCPU),AREAR(LIMCPU)
      COMMON /PROCSRS/ AREAL(LIMCPU),BOUNDR(LIMCPU),BOUNDL(LIMCPU)
      COMMON /PROCSRS/ LASSIGN(LIMCPU),IRETURN(LIMCPU),KQRETRN(LIMCPU)
      COMMON /PROCSRS/ DX(LIMCPU),XMID(LIMCPU),FMID(LIMCPU)
      COMMON /PROCSRS/ COTANR(LIMCPU),COTANL(LIMCPU),INFL(LIMCPU)
      DIMENSION TARRAY(2)
      DATA NFUN / 10,50,100,500,1000 /
      DATA NEPS / 10,100,1000,10000,100000 /
      DATA LEFT,CENTER,RIGHT / 1,2,3 /
      DATA A,B /0.,1./

      TIME1 = DTIME(TARRAY)
C
C     FOR-LOOP FROM 1 TO KASES
C
      DO 600 LOOP=1,KASES
C
C       INITIALIZES THE ALGORITHM
C
      AREA = 0.
      EPS = 1./NEPS(LOOP)
      DISCARD = EPS/(B-A)
C
C       FIND THE INITIAL INTERVAL LENGHT
C
      DX1 = 1.
      NQ = (B-A)/DX1
      IF (DX1*NQ .LT. B-A) NQ = NQ + 1
      DX1 = (B-A)/NQ
      LASTQ = NQ
      LEADER = 1
      COUNT = NQ
C
C       FIRST SET OF QUANTITIES FOR INITIAL INTERVALS
C
      DO 100 K = 1,NQ
         XRIGHT(K) = A + K*DX1
         XLEFT(K) = XRIGHT(K) - DX1
         FRIGHT(K) = F(XRIGHT(K))
         FLEFT(K) = F(XLEFT(K))
         AEST(K) = .5*DX1*(FLEFT(K) + FRIGHT(K))
         AREA = AREA + AEST(K)
         COTAN(K) = DX1/(FRIGHT(K) - FLEFT(K))
         IRIGHT(K) = K+1
         ILEFT(K) = K-1
         INEXT(K) = K+1
  100 CONTINUE
C
C       FIX ITEMS FOR END INTERVALS NOT SET CORRECTLY ABOVE
C
      ILEFT(1) =LIMQ
      IRIGHT(NQ) = LIMQ
      INEXT(NQ) = LIMQ
C
C       SECOND SET OF QUANTITIES FOR INITIAL INTERVALS
C
      COTL(1) = 0.
      COTR(NQ) = 0.
      INFLECT(1) = 0
      INFLECT(NQ) = 0
C
C       DETERMINE INTERVALS WHERE COTANGENT DIFFERENCES CHANGE SIGN
C
      IF (NQ .EQ. 1) GOTO 201
      COTR(1) = COTAN(2)
      COTL(NQ) = COTAN(NQ-1)
      IF (NQ .LE. 2) GOTO 201
      DO 200 K = 2,NQ-1
         COTL(K) = COTAN(K-1)
         COTR(K) = COTAN(K+1)
         IF (ABS(COTL(K)-COTR(K)) .EQ.
     *          (ABS(COTL(K)-COTAN(K)) + ABS(COTAN(K)-COTR(K)))) THEN
            INFLECT(K) = 0
         ELSE
            INFLECT(K) = CENTER
            INFLECT(K-1) = LEFT
            INFLECT(K+1) = RIGHT
         ENDIF
  200 CONTINUE
  201 CONTINUE
C
C       NOW COMPUTE THE INITIAL ERROR BOUND
C
      BOUNDA = 0.
      DO 300 K = 1, NQ
         IF (INFLECT(K) .EQ. CENTER) THEN
            COTREC = 1./COTR(K) + 1./COTL(K)
            BOUND(K) = DX1*ABS(FLEFT(K) - FRIGHT(K) + COTREC)
         ELSE
            DF = FRIGHT(K) - FLEFT(K)
            IF (INFLECT(K) .EQ. LEFT)
     *         BOUND(K) = TRIANGL(COTL(K), COTAN(K), 0, DX1, DF)
            IF (INFLECT(K) .EQ. RIGHT)
     *         BOUND(K) = TRIANGL(0, COTAN(K), COTR(K), DX1, DF)
            IF (INFLECT(K) .EQ. 0)
     *         BOUND(K) = TRIANGL(COTL(K), COTAN(K), COTR(K), DX1, DF)
         ENDIF
         BOUNDA = BOUNDA + BOUND(K)
  300 CONTINUE
C
C       FINALLY FREE ALL INTERVALS AND MARK THEM AS IN THE QUEUE
C
      DO 400 K = 1, NQ
         INQUEUE(K) = .TRUE.
  400 CONTINUE
      IQFREE = .TRUE.
      ITFREE = .TRUE.

      IP = 1
```

```
  500 IF (BOUNDA .LE. EPS) FINISH = 1.
      IF (NQ .GE. LIMQ) THEN
         XAREA(LOOP) = -1.
         FINISH = 0.
         GOTO 600
      ENDIF
      IF (FINISH .EQ. 1.) THEN
         XAREA(LOOP) = AREA
         XBOUNDA(LOOP) = BOUNDA
         FINISH = 0.
         GOTO 600
      ENDIF

      CALL GET(IP)
      CALL AREAS(IP)
      CALL PUT(IP)
      CALL INSERT(IP)
      GOTO 500
  600 CONTINUE
      TIME2 = DTIME(TARRAY)

      DO 900 LOOP=1,KASES
         PRINT 700,1./NEPS(LOOP),NFUN(LOOP)
  700    FORMAT ('PROBLEM 17 WITH EPS,NP = ',F10.8,5X,I10)
         IF (XAREA(LOOP) .EQ. -1.) THEN
            PRINT *, 'ABNORMAL STOP!'
         ELSE
            PRINT 800,XAREA(LOOP),XBOUNDA(LOOP)
  800       FORMAT ('GIVES THE AREA = ',F10.8,' AND THE BOUND = ',F10.8)
         ENDIF
  900 CONTINUE
      PRINT 950,TIME2,TARRAY(1),TARRAY(2)
  950 FORMAT ('TIME ',F8.4,' SECONDS, (',F8.4,' USER, ',F8.4,' SYSTEM)')
      STOP
      END


C
C-----------------------------------------------------------------------
C
      SUBROUTINE AREAS(IP)
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      COMMON /PROBLEM/ A,B,EPS
      COMMON /CONTROL/ AREA,BOUNDA,DISCARD,FINISH
      COMMON /QUEUE/ LEADER,NQ,LASTQ,INEXT(LIMQ),AEST(LIMQ),BOUND(LIMQ)
      COMMON /QUEUE/ INFLECT(LIMQ),COTAN(LIMQ),COTR(LIMQ),COTL(LIMQ)
      COMMON /QUEUE/ XRIGHT(LIMQ),FRIGHT(LIMQ),IRIGHT(LIMQ),XLEFT(LIMQ)
      COMMON /QUEUE/ FLEFT(LIMQ),ILEFT(LIMQ),INQUEUE(LIMQ),IQFREE,IDQ
      COMMON /QUEUE/ WAITING,NEXTQ,ITFREE,IDT,TAILING,NEXTT,COUNT
      COMMON /PROCSRS/ ACHANGE(LIMCPU),BCHANGE(LIMCPU),AREAR(LIMCPU)
      COMMON /PROCSRS/ AREAL(LIMCPU),BOUNDR(LIMCPU),BOUNDL(LIMCPU)
      COMMON /PROCSRS/ IASSIGN(LIMCPU),IRETURN(LIMCPU),KQRETRN(LIMCPU)
      COMMON /PROCSRS/ DX(LIMCPU),XMID(LIMCPU),FMID(LIMCPU)
      COMMON /PROCSRS/ COTANR(LIMCPU),COTANL(LIMCPU),INFL(LIMCPU)

      IAI = IASSIGN(IP)
C
C       PRELIMINARY QUANTITIES
C
      DX(IP) = .5*(XRIGHT(IAI) - XLEFT(IAI))
      XMID(IP) = XRIGHT(IAI) - DX(IP)
      FMID(IP) = F(XMID(IP))
      COTANR(IP) = ABS(DX(IP)/(FRIGHT(IAI)-FMID(IP)))
      COTANL(IP) = ABS(DX(IP)/(FMID(IP) - FLEFT(IAI)))
C
C       CHECK INTERVAL SITUATION AND SELECT AREA FORMULAS
C
      IF (INFLECT(IAI) .EQ. 0) THEN
         BOUNDL(IP) = TRIANGL(COTL(IAI),COTANL(IP),COTANR(IP),
     *                        DX(IP),(FLEFT(IAI) - FMID(IP)))
         BOUNDR(IP) = TRIANGL(COTANL(IP),COTANR(IP),COTR(IAI),
     *                        DX(IP),(FMID(IP) - FRIGHT(IAI)))
         INFL(IP) = 0
      ELSE
         CALL SPECIAL(COTL(IAI),COTANL(IP),COTANR(IP),COTR(IAI),
     *                INFLECT(IAI),IP)
      ENDIF
C
C       CHECK DISCARDING OF INTERVALS
C
      IRETURN(IP) = 2
      IF (BOUNDR(IP) .LT. DISCARD*DX(IP)) IRETURN(IP) = 1
      IF (BOUNDL(IP) .LT. DISCARD*DX(IP)) IRETURN(IP) = IRETURN(IP) - 1
C
C       COMPUTE CHANGES IN AREA AND BOUNDA
C
      AREAR(IP) = .5*DX(IP)*(FMID(IP) + FRIGHT(IAI))
      AREAL(IP) = .5*DX(IP)*(FMID(IP) + FLEFT(IAI))
      BCHANGE(IP) = BOUND(IAI) - BOUNDR(IP) - BOUNDL(IP)
      ACHANGE(IP) = AEST(IAI) - AREAR(IP) - AREAL(IP)
      RETURN
      END

C
C-----------------------------------------------------------------------
C
      FUNCTION TRIANGL(CLFT, CENT, CRGT, XBASE, YBASE)

      COTRGT = ABS((CRGT*CENT + 1.)/(CRGT - CENT))
      COTLFT = ABS((CENT*CLFT + 1.)/(CENT - CLFT))
      BASE2 = XBASE**2 + YBASE**2
      TRIANGL = ABS(.5*BASE2/(COTRGT + COTLFT))
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE SPECIAL(CLL, CL, CR, CRR, NFLECT, IP)
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      COMMON /PROBLEM/ A,B,EPS
      COMMON /CONTROL/ AREA,BOUNDA,DISCARD,FINISH
      COMMON /QUEUE/ LEADER,NQ,LASTQ,INEXT(LIMQ),AEST(LIMQ),BOUND(LIMQ)
      COMMON /QUEUE/ INFLECT(LIMQ),COTAN(LIMQ),COTR(LIMQ),COTL(LIMQ)
      COMMON /QUEUE/ XRIGHT(LIMQ),FRIGHT(LIMQ),IRIGHT(LIMQ),XLEFT(LIMQ)
      COMMON /QUEUE/ FLEFT(LIMQ),ILEFT(LIMQ),INQUEUE(LIMQ),IQFREE,IDQ
      COMMON /QUEUE/ WAITING,NEXTQ,ITFREE,IDT,TAILING,NEXTT,COUNT
      COMMON /PROCSRS/ ACHANGE(LIMCPU),BCHANGE(LIMCPU),AREAR(LIMCPU)
      COMMON /PROCSRS/ AREAL(LIMCPU),BOUNDR(LIMCPU),BOUNDL(LIMCPU)
      COMMON /PROCSRS/ IASSIGN(LIMCPU),IRETURN(LIMCPU),KQRETRN(LIMCPU)
      COMMON /PROCSRS/ DX(LIMCPU),XMID(LIMCPU),FMID(LIMCPU)
      COMMON /PROCSRS/ COTANR(LIMCPU),COTANL(LIMCPU),INFL(LIMCPU)
      REAL CHANGE(LIMCPU)
C
C       ARITHMETIC STATEMENT FUNCTIONS DETERMINE MONOTONICITY OF
C       COTANGENT SEQUENCE ON THREE POINTS
```

```
C
      CHANGE3(IP) = ABS(CLL-CRR) -
     *              (ABS(CLL-CL) + ABS(CL-CR) + ABS(CR-CRR))
      QUADRIL(CLFT, CRGT, DX, DF) = DX*ABS(DF + 1./CLFT + 1./CRGT)
      IAI = IASSIGN(IP)
      DFL = FMID(IP) - FLEFT(IAI)
      DFR = FRIGHT(IAI) - FMID(IP)
C
C       SELECT ONE OF THREE CASES FOR INFLECT
C
      IF (NFLECT .EQ. CENTER) GOTO 300
      IF (NFLECT .EQ. RIGHT) GOTO 200
C
C       INFLECT = LEFT
C
      IF (CHANG3(IP) .EQ. 0) THEN
         BOUNDL(IP) = TRIANGL(CLL,CL,CR,DX(IP),DFL)
         BOUNDR(IP) = TRIANGL(CL,CR,0.,DX(IP),DFR)
         INFL(IP) = 0
      ELSE
         BOUNDL(IP) = TRIANGL(CLL,CL,0.,DX(IP),DFL)
         BOUNDR(IP) = QUADRIL(CL,CRR,DX(IP),DFR)
         INFL(IP) = LEFT
         INFLECT(IAI) = CENTER
         INFLECT(IRIGHT(IAI)) = RIGHT
         INFLECT(IRIGHT(IRIGHT(IAI))) = 0
      ENDIF
      RETURN
C
C       INFLECT = RIGHT
C
  200 CONTINUE
      IF (CHANG3(IP) .EQ. 0) THEN
         BOUNDL(IP) = TRIANGL(0.,CL,CR,DX(IP),DFL)
         BOUNDR(IP) = TRIANGL(CL,CR,CRR,DX(IP),DFR)
         INFL(IP) = RIGHT
         INFLECT(IAI) = 0
      ELSE
         BOUNDL(IP) = QUADRIL(CRR,CL,DX(IP),DFL)
         BOUNDR(IP) = TRIANGL(0.,CR,CRR,DX(IP),DFR)
         INFL(IP) = CENTER
         INFLECT(IAI) = RIGHT
         INFLECT(ILEFT(IAI)) = LEFT
         INFLECT(ILEFT(ILEFT(IAI))) = 0
      ENDIF
      RETURN
C
C       INFLECT = CENTER
C
  300 CONTINUE
      IF (ABS(CL-CRR) .LT. (ABS(CL-CR) + ABS(CR-CRR))) THEN
         BOUNDL(IP) = TRIANGL(CLL,CL,0.,DX(IP),DFL)
         BOUNDR(IP) = QUADRIL(CL,CRR,DX(IP),DFR)
         INFLECT(ILEFT(IAI)) = 0
      ELSE
         BOUNDL(IP) = QUADRIL(CLL,CR,DX(IP),DFL)
         BOUNDR(IP) = TRIANGL(0.,CR,CRR,DX(IP),DFR)
         INFL(IP) = CENTER
         INFLECT(IAI) = RIGHT
         INFLECT(IRIGHT(IAI)) = 0
      ENDIF
      RETURN

      END
```

```
C
C-----------------------------------------------------------------------
C
      SUBROUTINE INSERT(IP)
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      COMMON /PROBLEM/ A,B,EPS
      COMMON /CONTROL/ AREA,BOUNDA,DISCARD,FINISH
      COMMON /QUEUE/ LEADER,NQ,LASTQ,INEXT(LIMQ),AEST(LIMQ),BOUND(LIMQ)
      COMMON /QUEUE/ INFLECT(LIMQ),COTAN(LIMQ),COTR(LIMQ),COTL(LIMQ)
      COMMON /QUEUE/ XRIGHT(LIMQ),FRIGHT(LIMQ),IRIGHT(LIMQ),XLEFT(LIMQ)
      COMMON /QUEUE/ FLEFT(LIMQ),ILEFT(LIMQ),INQUEUE(LIMQ),IQFREE,IDQ
      COMMON /QUEUE/ WAITING,NEXTQ,ITFREE,IDT,TAILING,NEXTT,COUNT
      COMMON /PROCSRS/ ACHANGE(LIMCPU),BCHANGE(LIMCPU),AREAR(LIMCPU)
      COMMON /PROCSRS/ AREAL(LIMCPU),BOUNDR(LIMCPU),BOUNDL(LIMCPU)
      COMMON /PROCSRS/ IASSIGN(LIMCPU),IRETURN(LIMCPU),KQRETRN(LIMCPU)
      COMMON /PROCSRS/ DX(LIMCPU),XMID(LIMCPU),FMID(LIMCPU)
      COMMON /PROCSRS/ COTANR(LIMCPU),COTANL(LIMCPU),INFL(LIMCPU)

      IL = IASSIGN(IP)
      IAI = IASSIGN(IP)
      IR = KQRETRN(IP)
C
C       CHECK ABOUT DISCARDING RIGHT INTERVAL
C
      IF (BOUNDR(IP) .LT. DISCARD*DX(IP)) THEN
         ILEFT(IRIGHT(IAI)) = LIMQ
         IR = LIMQ
         GOTO 200
      ENDIF
C
C       INSERT RIGHT INTERVAL INTO IAI = IL IF LEFT ONE IS DISCARDED
C
      IF (BOUNDL(IP) .LT. DISCARD*DX(IP)) THEN
         IR = IL
         IL = LIMQ
      ENDIF
C
C       INSERT RIGHT INTERVAL INFORMATION INTO THE COLLECTION
C
      XRIGHT(IR) = XRIGHT(IAI)
      FRIGHT(IR) = FRIGHT(IAI)
      XLEFT(IR) = XMID(IP)
      FLEFT(IR) = FMID(IP)
      BOUND(IR) = BOUNDR(IP)
      AEST(IR) = AREAR(IP)
      COTAN(IR) = COTANR(IP)
      ILEFT(IR) = IL
      IRIGHT(IR) = IRIGHT(IAI)
      COTR(IR) = COTR(IAI)
      COTL(IR) = COTANL(IP)
      INFLECT(IR) = INFLECT(IAI)
      INQUEUE(IR) = .TRUE.
C
C       CHECK ABOUT DISCARDING LEFT INTERVAL
C
  200 IF (BOUNDL(IP) .LT. DISCARD*DX(IP)) THEN
         IRIGHT(ILEFT(IAI)) = LIMQ
         GOTO 300
      ENDIF
C
C       INSERT LEFT INTERVAL INFORMATION INTO THE COLLECTION
C
```

```
      XRIGHT(IL) = XHID(IP)
      FRIGHT(IL) = FHID(IP)
      XLEFT(IL) = XLEFT(IAI)
      FLEFT(IL) = FLEFT(IAI)
      BOUND(IL) = BOUNDL(IP)
      AEST(IL) = AREAL(IP)
      COTAN(IL) = COTANL(IP)
      ILEFT(IL) = ILEFT(IAI)
      IRIGHT(IL) = IR
      COTR(IL) = COTANR(IP)
      COTL(IL) = COTL(IAI)
      INFLECT(IL) = INFL(IP)
      INQUEUE(IL) = .TRUE.
  300 RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE GET(IP)
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      COMMON /PROBLEM/ A,B,EPS
      COMMON /CONTROL/ AREA,BOUNDA,DISCARD,FINISH
      COMMON /QUEUE/ LEADER,NQ,LASTQ,INEXT(LIMQ),AEST(LIMQ),BOUND(LIMQ)
      COMMON /QUEUE/ INFLECT(LIMQ),COTAN(LIMQ),COTR(LIMQ),COTL(LIMQ)
      COMMON /QUEUE/ XRIGHT(LIMQ),FRIGHT(LIMQ),IRIGHT(LIMQ),XLEFT(LIMQ)
      COMMON /QUEUE/ FLEFT(LIMQ),ILEFT(LIMQ),INQUEUE(LIMQ),IQFREE,IDQ
      COMMON /QUEUE/ WAITING,NEXTQ,ITFREE,IDT,TAILING,NEXTT,COUNT
      COMMON /PROCSRS/ ACHANGE(LIMCPU),BCHANGE(LIMCPU),AREAR(LIMCPU)
      COMMON /PROCSRS/ AREAL(LIMCPU),BOUNDR(LIMCPU),BOUNDL(LIMCPU)
      COMMON /PROCSRS/ IASSIGN(LIMCPU),IRETURN(LIMCPU),KQRETRN(LIMCPU)
      COMMON /PROCSRS/ DX(LIMCPU),XHID(LIMCPU),FHID(LIMCPU)
      COMMON /PROCSRS/ COTANR(LIMCPU),COTANL(LIMCPU),INFL(LIMCPU)

      IF (LEADER .EQ. LIMQ) THEN
         RETURN
      ELSE
         IASSIGN(IP) = LEADER
         INQUEUE(IP) = .FALSE.
         LEADER = INEXT(LEADER)
      ENDIF
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE PUT(IP)
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      COMMON /PROBLEM/ A,B,EPS
      COMMON /CONTROL/ AREA,BOUNDA,DISCARD,FINISH
      COMMON /QUEUE/ LEADER,NQ,LASTQ,INEXT(LIMQ),AEST(LIMQ),BOUND(LIMQ)
      COMMON /QUEUE/ INFLECT(LIMQ),COTAN(LIMQ),COTR(LIMQ),COTL(LIMQ)
      COMMON /QUEUE/ XRIGHT(LIMQ),FRIGHT(LIMQ),IRIGHT(LIMQ),XLEFT(LIMQ)
      COMMON /QUEUE/ FLEFT(LIMQ),ILEFT(LIMQ),INQUEUE(LIMQ),IQFREE,IDQ
      COMMON /QUEUE/ WAITING,NEXTQ,ITFREE,IDT,TAILING,NEXTT,COUNT
      COMMON /PROCSRS/ ACHANGE(LIMCPU),BCHANGE(LIMCPU),AREAR(LIMCPU)
      COMMON /PROCSRS/ AREAL(LIMCPU),BOUNDR(LIMCPU),BOUNDL(LIMCPU)
      COMMON /PROCSRS/ IASSIGN(LIMCPU),IRETURN(LIMCPU),KQRETRN(LIMCPU)
      COMMON /PROCSRS/ DX(LIMCPU),XHID(LIMCPU),FHID(LIMCPU)
      COMMON /PROCSRS/ COTANR(LIMCPU),COTANL(LIMCPU),INFL(LIMCPU)

      IF (IRETURN(IP) .EQ. 0) THEN
         COUNT = COUNT - 1
C
C       NO INTERVALS RETURNED
C
      ENDIF
      IF (IRETURN(IP) .EQ. 1) THEN
C
C       PICK UP INFO TO PUT NEW INTERVAL IN OLD PLACE
C
         INEXT(NQ) = IASSIGN(IP)
         NQ = IASSIGN(IP)
         INEXT(NQ) = LIMQ
      ENDIF
      IF (IRETURN(IP) .EQ. 2) THEN
C
C       PICK UP INFO TO PUT 1 NEW INTERVAL IN OLD PLACE AND
C       EXTEND QUEUE AREA BY 1 FOR THE OTHER NEW INTERVAL
C
         LASTQ = LASTQ + 1
         INQUEUE(LASTQ) = .FALSE.
         KQRETRN(IP) = LASTQ
         INEXT(IASSIGN(IP)) = LASTQ
         IF (COUNT .GT. 1) INEXT(NQ) = IASSIGN(IP)
         NQ = LASTQ
         INEXT(NQ) = LIMQ
         COUNT = COUNT + 1
      ENDIF
C
C       REASSIGN THE QUEUE LEADER IF THE QUEUE WAS EMPTY
C
      IF ((IRETURN(IP) .GT. 0) .AND. (LEADER .EQ. LIMQ)) THEN
         LEADER = IASSIGN(IP)
      ENDIF
C
C       UPDATE TEH AREA AND BOUND ESTIMATES
C
      AREA = AREA - ACHANGE(IP)
      BOUNDA = BOUNDA - BCHANGE(IP)

      RETURN
      END

      REAL FUNCTION F (X)
      PARAMETER (LIMQ=100000,LIMCPU=1,KASES=5)
      COMMON /ALGORITHM/ LOOP,NFUN(KASES),NEPS(KASES)
      COMMON /ALGORITHM/ XAREA(KASES),XBOUNDA(KASES)
      F = 1.
      DO 10 I = 1, NFUN(LOOP)
         F = COS(F*SIN(X))*SIN(X*SIN(X+F))
   10 CONTINUE
      RETURN
      END
```